Spring August 2014

Unsupervised Joint Alignment, Clustering and Feature Learning

Mohamed Marwan Mattar
University of Massachusetts - Amherst

UNSUPERVISED JOINT ALIGNMENT, CLUSTERING AND FEATURE LEARNING

A Dissertation Presented

by

MOHAMED MARWAN ABDEL MAGID MATTAR

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2014

School of Computer Science

UNSUPERVISED JOINT ALIGNMENT, CLUSTERING AND FEATURE LEARNING

A Dissertation Presented

by

MOHAMED MARWAN ABDEL MAGID MATTAR

Approved as to style and content by:

Allen R. Hanson, Co-chair

Erik G. Learned-Miller, Co-chair

Benjamin M. Marlin, Member

John W. Stuadenmayer, Member

Lori A. Clarke, Chair
School of Computer Science

In loving memory of my grandfather, Abdel Aal Ibrahim Abdel Aal.

ACKNOWLEDGMENTS

I am incredibly fortunate to have had the wonderful support of advisors, colleagues, friends, and family throughout my education. Spending a short eleven years at UMass Amherst has been a surreal experience, and I am grateful for all the friends I've made over the years who make Amherst a very special place for me.

First off, a huge thanks to Allen and Erik for advising me on all three degrees. From my early days in the computer vision lab I was blown away by the amount of freedom and support I was given to learn and explore. Allen, it took the patience of a wizard such as yourself to support me through these years and I am incredibly grateful for letting me roam free and work on whatever tickled my interests. You created the perfect environment where mistakes were never judged, but treated as learning experiences, and risks were encouraged. I can't thank you enough for the positive impact you've had on me, both personally and professionally. Erik, thank you for being there during every step of my journey, which started with our NIPS trip in 2004 and our subsequent curve congealing project. I could always count on you to offer advice and guidance when I needed it. Your sense of humor (yes, I'm finally admitting it) is a nice plus too. Your backhand swing still needs some work though.

A special thanks to the rest of my thesis committee. Ben, I was ecstatic to have you on-board and benefited greatly from our discussions. You get the most points for impact per unit time. John, taking four statistics courses with you was the single best thing I did for my thesis. Thank you for all the hard work you put into your teaching and for the great insights you've offered on this thesis. You are a prime example that excellent researchers can also be great teachers.

The hopes and dreams of a graduate student typically hinge on the mood swings of anonymous reviewers during conference deadlines. So having a strong support group to amplify the wins and soften the blows is necessary to maintain sanity. A big thank you to friends and colleagues who made my graduate school experience an enjoyable and memorable one: John D'Agostino, Luke Lysak, Dima Lisin, Frank Stolle, Matthew Blaschko, Pla Silapachote, Gary Holness, Jerod Weinman, Paul Dickson, Walker Duhon, Adam Williams, Vidit Jain, Ben Mears, David Smith, Evan Shelhamer, Armita Kaboli, Bobby Simidchieva, Katerina Marazopoulou, Priti Raghuvir, Sameer Singh, Marianne Seney, Jeff Sabelli, Saber Zohir, Kit Fuderich, Stephanie Tenerowicz, Michael Ross, Howard Schultz, Mark Benfield, Ben Tupper, Mike Sierarcki, Edward Riseman, and Paul Utgoff.

A special thanks to Daniel Gyllstrom for being the best of friends and the most chilled out entertainer I know; to Gary Huang for being the best collaborator; and to the 2014 vision lab graduates (Andrew Kae, Jackie Feild, Manju Narayana, and Laura Sevilla-Lara) for all their friendship and support, especially during the final hours. I couldn't have asked for a better group of labmates.

A big thanks to the computer science staff for all their support over the years. I am particularly indebted to CSCF for their technical support, to Laurie Downey for her administrative assistance, and to Leanne Leclerc for the insurmountable task of making sure I completed the milestones on-time.

Lastly, my parents and brother have been a constant source of love, support, wisdom and comfort. They ingrained in me the importance of intellectual curiosity and its pursuit, and always encouraged me to do what I loved. Without them, this thesis would not have been possible.

ABSTRACT

UNSUPERVISED JOINT ALIGNMENT, CLUSTERING AND FEATURE LEARNING

MAY 2014

MOHAMED MARWAN ABDEL MAGID MATTAR

B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Allen R. Hanson and Professor Erik G. Learned-Miller

Joint alignment is the process of transforming instances in a data set to make them more similar based on a pre-defined measure of joint similarity. This process has great utility and applicability in many scientific disciplines including radiology, psychology, linguistics, vision, and biology. Most alignment algorithms suffer from two shortcomings. First, they typically fail when presented with complex data sets arising from multiple modalities such as a data set of normal and abnormal heart signals. Second, they require hand-picking appropriate feature representations for each data set, which may be time-consuming and ineffective, or outside the domain of expertise for practitioners.

In this thesis we introduce alignment models that address both shortcomings. In the first part, we present an efficient curve alignment algorithm derived from the congealing framework that is effective on many synthetic and real data sets. We show

that using the byproducts of joint alignment, the aligned data and transformation parameters, can dramatically improve classification performance. In the second part, we incorporate unsupervised feature learning based on convolutional restricted Boltzmann machines to learn a representation that is tuned to the statistics of the data set. We show how these features can be used to improve both the alignment quality and classification performance. In the third part, we present a nonparametric Bayesian joint alignment and clustering model which handles data sets arising from multiple modes. We apply this model to synthetic, curve and image data sets and show that by simultaneously aligning and clustering, it can perform significantly better than performing these operations sequentially. It also has the added advantage that it easily lends itself to semi-supervised, online, and distributed implementations.

Overall this thesis takes steps towards developing an unsupervised data processing pipeline that includes alignment, clustering and feature learning. While clustering and feature learning serve as auxiliary information to improve alignment, they are important byproducts. Furthermore, we present a software implementation of all the models described in this thesis. This will enable practitioners from different scientific disciplines to utilize our work, as well as encourage contributions and extensions, and promote reproducible research.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **vii**

LIST OF TABLES . . . . . **xi**

LIST OF FIGURES . . . . . **xiii**

CHAPTER

1. INTRODUCTION **1**

1.1 Interlude: Alignment is ill-posed 3
1.2 Previous Work 4
1.3 Contributions and Outline 6

2. CURVE ALIGNMENT USING CONGEALING **10**

2.1 Congealing Overview 10
2.2 Algorithm 13
2.3 Experimental Results 18

2.3.1 Evaluation using Synthetic Data Sets 19
2.3.2 Evaluation using Real Data Sets 24
2.3.3 Improving Classification with Unsupervised Alignment 25

2.3.3.1 Interlude: Dynamic Time Warping 29
2.3.3.2 Interlude: Multiple Kernel Learning 30
2.3.3.3 Experimental Results 31

2.4 Discussion and Drawbacks 35

3. INCORPORATING FEATURE LEARNING **38**

3.1 Background: Convolutional RBM 40

3.2 Congealing with CRBMs . . . 43
3.2.1 Feature Learning . . . 43
3.2.2 Alignment Objective Function . . . 46
3.3 Experiment: Curve Alignment using CRBMs . . . 47
3.4 Experiment: Face Alignment with CRBMs . . . 51
3.5 Summary . . . 52
4. JOINT ALIGNMENT AND CLUSTERING . . . 60
4.1 Bayesian Joint Alignment . . . 62
4.1.1 Learning . . . 63
4.1.2 Model Characteristics . . . 66
4.1.3 Experiments . . . 67
4.2 Clustering Extension using Bayesian Nonparametrics . . . 69
4.2.1 Learning . . . 71
4.2.2 Incorporating Labelled Examples . . . 75
4.2.3 Experiment: Alignment and Clustering of Digits . . . 75
4.2.4 Experiment: Alignment and Clustering of Curves . . . 80
4.3 Large-scale Learning . . . 81
4.3.1 Online Learning . . . 81
4.3.2 Distributed Learning . . . 82
4.4 Summary . . . 83
5. SOFTWARE AND REPRODUCIBLE RESEARCH . . . 85
6. CONCLUSION AND SUMMARY . . . 90
BIBLIOGRAPHY . . . 92

LIST OF TABLES

Table **Page**

2.1 Pseudo-code for our stochastic curve congealing algorithm. 16

2.2 Curve classification using a linear SVM. The first column contains the name of the UCR data set (one per row) and the remaining eight columns contain the classification performance using different techniques. Column 2: direct classification using a linear SVM classifier, without alignment. Column 3: classification using dynamic time warping (DTW). Columns 4 - 9: classification based on joint alignment. We experimented with 6 different settings which are indicated by the three rows labeled "Objective" (entropy vs. variance), "Trans." (all transformations vs. nonlinear time scaling), and "Grouping" (simple concatenation vs. multiple kernel learning). 31

2.3 Curve classification using a second-degree polynomial SVM. The first column contains the name of the UCR data set (one per row) and the remaining five columns contain the classification performance using different techniques. Column 2: direct classification using a second-degree polynomial SVM classifier, without alignment. Column 3: classification using dynamic time warping (here, we include results reported on the UCR time series repository webpage). Columns 4 - 6: classification based on joint alignment. We experimented with 3 different settings which are indicated by the three rows labeled "Objective" (entropy vs. variance), "Trans." (all transformations vs nonlinear time scaling), and "Grouping" (simple concatenation). 34

3.1 Top-level pseudo-code for learning a convolutional RBM. 44

3.2 Curve classification using a linear SVM. The first column contains the name of the UCR data set (one per row) and the remaining five columns contain the classification performance using different settings which are indicated by the rows labeled "Alignment" and "Classification" which specify the feature representation used for each task. For example, column 5 contains the classification accuracies when alignment was performed using the CRBM features and classification was performed using the raw curves. For all the experiments that include alignment (columns 4 - 6), the input to the SVM classifier was the concatenation of the original curve, the transformed curve, and the transformation parameters (i.e. the simple concatenation method used in Chapter 2). 49

4.1 Single-class digit alignment. We evaluated three techniques: Original (no alignment), Congealing (binary congealing [49]), and our proposed Bayesian alignment model. We computed two alignment scores: the mean entropy across each pixel stack, and the mean pairwise distance across all the images. 67

4.2 Joint alignment and clustering of images. The top subtable refers to the first digit data set comprising 100 images of the digits "4" and "9" (Figure 4.1), while the bottom subtable refers to the second data set comprising 200 images of all 10 digits (the same data set used by Liu *et al.* [59], Figure 4.7). The alignment score columns contain three metrics that adhere to the following template: mean (standard deviation) ± standard error. The number following the clustering accuracy in the "Affinity propagation", "Infinite mixture model" and "Unsupervised JAC" rows is the number of clusters that the model discovered (i.e. chose to represent the data with). For both unsupervised and semi-supervised JAC we include the alignment score from the Gibbs sampler and after the cluster-specific post-processing (see text for more details). On both data sets, our models significantly outperform previous nonparametric alignment [49], clustering [20, 18], and joint alignment and clustering [21, 59]. 77

4.3 Joint alignment and clustering of ECG data. The alignment score columns contain three metrics that adhere to the following template: mean (standard deviation) ± standard error. The number following the clustering accuracy in the "Affinity propagation", "Infinite mixture model" and "Unsupervised JAC" rows is the number of clusters that the model discovered (i.e. chose to represent the data with). 80

LIST OF FIGURES

Figure **Page**

1.1 Ten audio signals from ten different individuals speaking the same phrase. The x-axis is time. **Left**: The original signals as recorded by a microphone. **Right:** The signals after they have been aligned using the continuous profile model [58]. 2

2.1 Alignment of binary images using congealing [66]. **Left:** before. **Right:** after. 11

2.2 Alignment of complex images using congealing [34]. **Left:** images of cars with a red bounding box representing the canonical location, orientation, and scale. **Right:** Similarly, for images of faces. 12

2.3 Bias removal result using congealing [50]. **Left:** before. **Right:** after. 12

2.4 Alignment of 3D brain volumes using congealing [103]. **Left:** 2D slices before alignment. **Right:** 2D slices after alignment. 13

2.5 The five original curves used to generate the 75 synthetic data sets. 19

2.6 Alignment results on two synthetic data sets. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. **Top row:** Synthetic data set generated with amplitude translation. **Bottom row:** Synthetic data set generated with linear scaling. For both data sets, the curves were perfectly aligned and are hidden behind the mean curve. 20

2.7 Alignment results on synthetic data sets generated with nonlinear time warping. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. The last row contains the worst alignment result for this transformation group. 21

2.8 Alignment results on synthetic data sets generated with nonlinear amplitude warping. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. The last row contains the worst alignment result for this transformation group. 22

2.9 Alignment results on synthetic data sets generated with all four transformations. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. The last row contains the worst alignment result for this transformation group. 23

2.10 Histogram of the alignment scores across all 75 synthetic data sets. The alignment score is the average pairwise Euclidean distance between the aligned curves. 24

2.11 Alignment of the Total Ion Count data set [58] using congealing. **Top**: original. **Bottom-left**: aligned with variance. **Bottom-right**: aligned with entropy. 25

2.12 Alignment of the ECG200 data set (both classes) using congealing with entropy. **Left:** original curves before alignment, color-coded by class. **Right:** aligned curves, also color-coded by class. The black curve in each plot represents the mean curve. 26

2.13 Alignment of the GunPoint data set (both classes) using congealing with variance. **Left:** original curves before alignment, color-coded by class. **Right:** aligned curves, also color-coded by class. The black curve in each plot represents the mean curve. 27

2.14 Alignment of the FaceFour data set (four classes) using congealing with variance. The **top row** displays the curves from all four classes before (left, blue) and after (right, red). The **middle** and **bottom** rows contain the curves in each of the four classes *after* alignment. The black curve in each plot represents the mean curve. 28

2.15 Classification scatter plots comparing two techniques. **Left:** Compares our best result (all transformations + variance + MKL) to the linear SVM baseline (no alignment). **Right:** Compares our best result (all transformations + variance + MKL) to dynamic time warping (DTW). 33

2.16 Classification scatter plots comparing two techniques. **Left:** Compares our best polynomial SVM result (all transformations + variance) to the polynomial SVM baseline (no alignment). **Right:** Compares our best polynomial SVM result (all transformations + variance) to our best linear SVM concatenation result (all transformation + variance). 35

2.17 Effect of congealing a multi-modal data set of 1's and 7's. The **first row** contains the mean images of the original data set, the **second row** contains the mean images after congealing the entire data set, and the **third row** shows the mean images after congealing each of the digit classes separately. 36

3.1 Boltzmann machines. **Left**: Standard RBM with 7 visible units and 4 hidden units. The graph is bipartite with dense (undirected) connections from every visible unit to every hidden unit. Each hidden unit is given a unique color that matches the color of its connections to the visible layer. Each edge between a hidden unit, h_i, and visible unit, v_j, has its own weight, w_{ij}. In this example, the weight matrix, W, is 4×7. **Right**: Max-pooling convolutional RBM also with 7 visible units and a single group (1 hidden layer consisting of 4 hidden units, and 2 pooling units). The superscript 1 for both the hidden and pooling nodes is to make explicit that they belong to the first group (the only group in this example). Again, each hidden unit is given a unique color that matches the color of its connections to the visible layer. However, unlike an RBM, each hidden unit is connected to contiguous (local) subset of the visible units and the weights of those connections are shared across all the hidden units. In this example, the following sets of connections share the same weight (the superscript is removed to make the notation less cumbersome): $\{(h_1, v_1), (h_2, v_2), (h_3, v_3), (h_4, v_4)\}$, $\{(h_1, v_2), (h_2, v_3), (h_3, v_4), (h_4, v_5)\}$, $\{(h_1, v_3), (h_2, v_4), (h_3, v_5), (h_4, v_6)\}$, $\{(h_1, v_4), (h_2, v_5), (h_3, v_6), (h_4, v_7)\}$. Thus, this CRBM is defined by 4 pairwise weights. Each additional group would have its own set of weights and would contribute an additional 4 parameters. Thus the weight matrix for a CRBM is $N_H \times K$ where N_H is the length of the filters (number of visible units that each hidden unit is connected to) and K is the number of groups. In this example, $N_H = 4$ and $K = 1$. 41

3.2 The 32 CRBM filters learned using 2700 curves from the UCR time series repository. The filter size is 10 and the pooling area is 5. The filters are ordered (row-major order) based on their mean pooling unit activations across all the curves (high mean activations first). 48

3.3 Classification scatter plots comparing a linear SVM without alignment or feature learning (column 2 in Table 3.2) to using both alignment and classification with CRBM representation (column 6 in Table 3.2). 50

3.4 Filters learned from a CRBM on LFW images. 51

3.5 CRBM pooling unit activations (the colors follow a standard scaled-image range where dark blue represents low activations and dark red represents high activations). For both plots, the filters are ordered from left to right by the magnitude of their activations (same ordering as in Figure 3.2). **Top:** Displays a matrix of the maximum pooling activation (for each of the 32 filters) on 50 randomly selected curves from each of the 13 data sets. The curves from each data set are grouped together and separated by a white line. **Bottom:** Similar to top, but we average the maximum pooling activation for each filter across all the curves in a data set. This provides us with a measure for how often a specific filter activates for each data set. 53

3.6 CRBM filters (1 - 8) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation). 54

3.7 CRBM filters (9 - 16) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation). 55

3.8 CRBM filters (17 - 24) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation). 56

3.9 CRBM filters (25 - 32) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation). 57

3.10 Clusters 1-10 of 20 resulting from running KMeans on the 3443 training images using the CRBM representation. Each row contains the eight images from each cluster that are closest (in Euclidean distance) to the cluster centroid. 58

3.11 Clusters 11-20 of 20 resulting from running KMeans on the 3443 training images using the CRBM representation. Each row contains the eight images from each cluster that are closest (in Euclidean distance) to the cluster centroid. 59

4.1 Joint alignment and clustering: given 100 unlabeled images (top), without any other information, our algorithm (§ 4.2) chooses to represent the data with two clusters, *aligns* the images and *clusters* them as shown (bottom). Our clustering accuracy is 94%, compared to 54% with K-means using two clusters (using the minimum error across 200 random restarts). Our model is not limited to affine transformations or images. 61

4.2 Illustrative example. (1) shows a data set of 2D points. The set of allowable transformations is rotations around the origin. (2) shows the result of the congealing algorithm which transforms points to minimize the sum of the marginal entropies. This independence assumption in the entropy computation causes the points to be squeezed into axis aligned groups. (3) highlights that clustering alone with an infinite mixture model may result in a larger number of clusters. (4) shows the result of the model presented in this chapter. It discovers two clusters and aligns the points in each cluster correctly. 62

4.3 Graphical representation for our proposed Bayesian alignment model (§ 4.1) 64

4.4 Comparison of congealing and Bayesian alignment on digits. Each image is the mean image of a specific digit class. **Top:** before alignment. **Middle:** alignment with congealing. **Bottom:** alignment with Bayesian alignment. Note that the qualitative differences between congealing and Bayesian alignment are negligible. Table 4.1 contains quantitative metrics. 66

4.5 Comparison of congealing and Bayesian alignment on curve data sets. **Top:** scatter plot of the mean pairwise distance score of congealing and the Bayesian alignment algorithm across the 75 synthetic curve data sets (generated in Chapter 2). **Middle and Bottom:** Two of the examples from Chapter two were congealing got stuck in local minima. 68

4.6 Graphical representation for our proposed nonparametric Bayesian joint alignment and clustering model (left) and its corresponding distributional form (right). 71

4.7 Unsupervised joint alignment and clustering of 200 images of all 10 digits. (Top) All 200 images provided to our model. (Bottom) The 11 clusters discovered and their alignments. 76

4.8 Joint alignment and clustering of ECG heart data (two classes, color-coded: red for abnormal, and blue for normal). The **first row** displays the original data set (left) and the result of congealing with entropy (left) and congealing with variance (right). The **last two rows** display the 5 clusters discovered by our unsupervised model. 84

5.1 Layout of the base classes included in our software package. 88

5.2 The visCongealing class extends the visAlignment abstract class. 88

5.3 The visBayesianAlignment class extends the visAlignment abstract class and uses two visSufficientStats objects to represent the data being aligned and the transformation parameters. 88

5.4 The visDPClustering class (which implements infinite mixture models) extends the visMixtureModel class to allow the creation and removal of classes within the sampling scheme. 89

5.5 The visAlignmentClustering extends both the visAlignment and visMixtureModel classes to allow the alignment and clustering of instances. Matlab allows multiple-inheritance. 89

CHAPTER 1

INTRODUCTION

Joint alignment is the process in which data points [1] are transformed to appear more similar to each other, based on a criterion of joint similarity. The purpose of alignment is typically to remove unwanted variability in a data set, by allowing the transformations that reduce that variability. See Figure 1.1 for an example of joint alignment of 10 real audio signals [2]. In the simplest and most reoccurring case this variability is noise, such as measurement noise. Consider the scenario where several individuals speak the same phrase into a microphone. Although there is one true underlying signal (that we care about) representing the audio of the spoken phrase, each signal has been independently transformed due to each individual's tone and pitch (amongst other things).

The purpose of alignment is to tease out this unwanted variability by allowing certain transformations on the data. This is either helpful in terms of recovering the true signal we care about or obtaining a representation that is invariant to factors we do not wish to model (i.e. a transformation-invariant representation). The process of alignment is widely applicable in a variety of domains. For example, removing temporal variability in event-related potentials allows psychologists to better localize brain responses [96], removing bias in magnetic resonance images provides doctors

---

[1] In this thesis, we analyze and present algorithms that can operate on multiple data types (e.g. images, curves, 3D volumes) and thus we use the generic term *data point* to emphasize this flexibility. In this sense, a *data point* can be an image, a curve or 3D volume and a *data set* is a collection of similar data points, such as binary images of digits or 3D brain volumes.

[2] Taken from `http://www.cs.toronto.edu/~jenn/alignmentStudy/Slides1_14.pdf` [58].

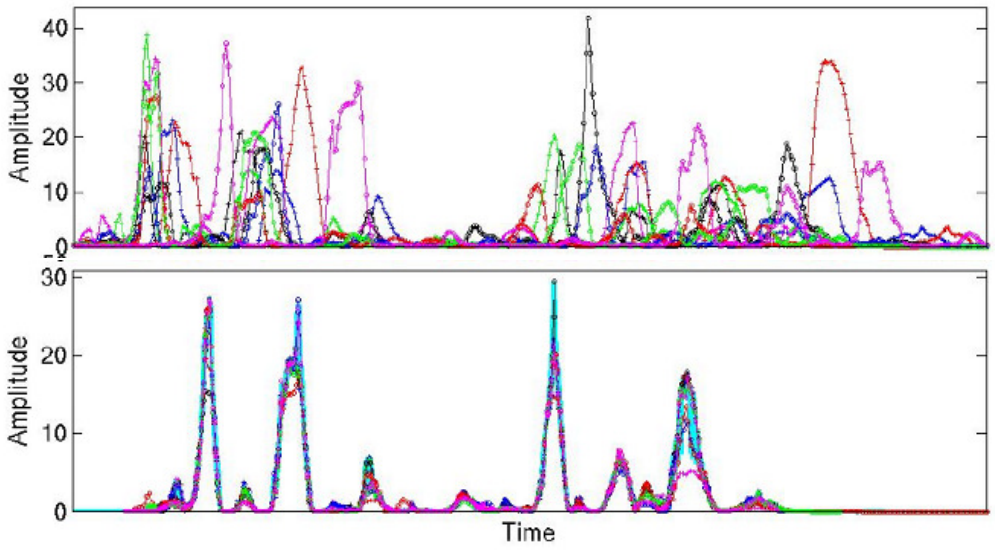

Figure 1.1: Ten audio signals from ten different individuals speaking the same phrase. The x-axis is time. **Left**: The original signals as recorded by a microphone. **Right:** The signals after they have been aligned using the continuous profile model [58].

with cleaner images for their analyses [49], and removing (affine) spatial variability in images of objects can improve the performance of joint compression [17] and recognition [34] algorithms. Specifically, it has been found that using an aligned version of the Labeled Faces in the Wild [36] data set significantly increases recognition performance [12], even for algorithms that explicitly handle misalignments. Consequently, the majority of results reported on the Labeled Faces in the Wild data set [3] use a version of the data set where the images have been aligned in an unsupervised manner [34, 37]. Aside from bringing data into correspondence, the process of alignment can be used for other scenarios. For example, if the data are similar up to known transformations, joint alignment can remove this variability and, in the process, recover the underlying latent data [58]. Also, the resulting transformations from alignment have been used to build classifiers using a single training example [49] and learn sprites in videos [42].

[3] `http://vis-www.cs.umass.edu/lfw/results.html`

1.1 Interlude: Alignment is ill-posed

It is important to note that joint alignment is an ill-posed problem; for two main reasons:

- The ambiguity of the latent data. Consider a data set generated from a single latent process, indicating that the aligned data should be identical to each other. There are several ways to transform the data such that the aligned data are identical to each other. While each such alignment will seem "perfect" in isolation, the reality is that they each hypothesize a *different* latent process. Since, typically, we are unaware of the true underlying process, we have no mechanism for accessing the relative accuracy of these seemingly "perfect" alignments.

- The ambiguity of the allowable transformations. The set of allowable transformations that an alignment algorithm can inflict on the data needs to be selected with care. If the transformation set is too flexible, it can allow the algorithm to *create* data, that is to transform the data in an unnatural fashion so as to "improve" alignment.

These ambiguities complicate the evaluation and comparison of alignment algorithms. In practice, the quality of an alignment is problem-specific and the choice of an alignment algorithm depends on its measure of similarity and how it agrees with what domain experts consider a good alignment. Despite these issues, alignment algorithms have been useful in a variety of applications where the alignments are not evaluated by an expert (§ 1.2 contains specific examples). Even alignment models that have a large capacity for warping (and creating) data can be useful. For example, Figure 1.1 shows audio signals after alignment with the *continuous profile model* (CPM) [58]. At first glance it may seem that this algorithm has created data. Curvatures in the original signals are removed, for example. However, the reality is that the audio from the combined aligned curves sounds much clearer than the audio

from the combined original curves [4]. This indicates that there are scenarios were completely warping the signal is useful and sometimes necessary.

1.2 Previous Work

Typically what distinguishes joint alignment algorithms are the assumptions they make about the data to be aligned and the transformations they can incur along with the level of supervision needed. Supervision takes several forms and can range from manually selecting landmarks to be aligned [12] to providing examples of data transformations [74]. In this thesis we focus on unsupervised joint alignment which is helpful in scenarios where supervision is not practical or available. Several such algorithms exist.

The first class of alignment algorithms were based on procrustes analysis [78], whose origins date back to Mosier in 1939 [67]. Procrustes analysis (a term coined by Hurley and Cattel in 1962 [39]) refers to the process of computing the optimal matching between two configurations of corresponding N-dimensional points under some transformation of one of the configurations. The residual sum-of-squares after matching is called the procrustes statistic. This process was first used for pairwise alignment and then generalized to joint alignment of an entire data set [46, 28]. The generality of procrustes analysis is evident by its application to an eclectic set of problems including shape analysis [43], multidimensional scaling [86], curve registration [77], manifold alignment [95], and learning binary codes for image representation [26].

A more recent flexible framework for joint alignment is congealing [49] which makes few assumptions about the data and allows the use of continuous transformations. It is a conditional maximization [23] optimization procedure that searches for the transformations parameters that maximize the probability of the aligned data

[4] `http://www.cs.toronto.edu/~jenn/alignmentStudy/`

under a kernel density estimate. Maximizing the likelihood is achieved by minimizing the entropy of the transformed data. It was initially applied to binary images of digits [66], but has since been extended to binary images of Drosophila discs [2], one-dimensional curves [64] [5], bias removal in MRI scans [50, 51], grayscale images of complex objects [34, 37], 3D brain volumes [103], and facial contour labeling [60]. Additionally, several congealing variants [93, 92, 14] have been presented that can improve its performance on binary images of digits and grayscale images of faces.

There are also several alignment algorithms that are specific to certain domains. For example, in the curve domain, the CPM [58] uses a variant of the hidden Markov model to locally transform each observation, while a mixture of regression model appended with global scale and translation transformations can simultaneously align and cluster [22]. In the image domain, the transformed mixture of Gaussians [21] and the work of Lui *et al.* [59] are used to align and cluster. In some cases, alignment algorithms are developed to handle a specific type of data. One example is event-related potential signals [62] where aligning signals across multiple subjects can recover an effective estimate of the stimulus offset [96, 41, 29].

One of the attractive properties of congealing is a clear separation between the transformation operator and optimization procedure. This has allowed congealing to be applied to a wide range of data types and transformation functions. Its main drawback is its inability to handle complex data sets that may contain multiple modes (i.e. images of the digits '1' and '7'). While congealing's use of an entropy-based objective function can in theory allow it to align multiple modes, in practice the independence assumption (temporally for curves and spatially for images) can cause it to collapse modes (see Figures 2.17 and 4.2 for an illustration). Additionally, its method for regularizing parameters to avoid excessive transformations is *ad hoc*

[5] Part of this thesis, see Chapter 2.

and does not prevent it from annihilating the data (shrinking to size zero) in some scenarios.

Several of the other methods presented above make many assumptions about the form of the data set or the transformations they it can incur. CPM assumes a single underlying curve. The mixture of regression model assumes the number of clusters in known a priori and only allows linear scaling and translation transformations. The transformed mixture of Gaussians also requires the number of clusters and is typically limited to small data sets and simple discrete transformations. This is due to its formulation which represents transformations discretely as matrices and every transformation parameter *value* [6] has an associated matrix.

1.3 Contributions and Outline

The goal of this thesis is to develop alignment models that are more general (i.e. can operate on curves, images or 3D data sets) and more effective, specifically on complex data sets. Complexities can arise in two forms:

1. Data sets that contain different groups of instances (multi-modal), such as an image data set of digits '1' and '7' or a curve data set of normal and abnormal heart signals. Typically, the correct number of groups is not known a priori.

2. Data sets that contain instances with high frequency structure. Intuitively, this refers to data sets of cars or faces, which are arguably more "complex" than data sets of the handwritten digit '0'. Formally, complexity in this case can be defined as the average pairwise Euclidean distance between instances in the data set if it were perfectly aligned. Note that this definition is dependent on the class of transformations used, so our claim that a data set of cars is more complex than a data set of the digit '0' is correct under affine transformations.

[6] For an NxN image one would need to use N^2 matrices to represent translations alone.

Another way to view this complexity is by comparing the variability in the data set to the variability spanned by the transformation operator. Digits, for example, largely contain only affine variability, while a data set of cars contains additional variability due to inherent differences in cars as well as projective transformations (mapping the 3D object onto a 2D image).

We propose to address the above-mentioned complexities through clustering and unsupervised feature learning, respectively. Clustering addresses the issue of multi-modal data sets, where instances are assigned to groups and aligned to members of their group. Unsupervised feature learning addresses the issue of data representation for complex data sets, by learning a representation from the ground-up that is specific to each data set. Specifically, we propose the following contributions:

1. Adapt the congealing framework for alignment of curve data sets (Chapter 2). This will enable to us to explore effective global parameterizations of curve transformations that will be utilized in subsequent work. This will also give us baseline alignment results using congealing in the curve domain to compare our proposed models to. We evaluate our congealing adaptation on both alignment and classification tasks using synthetic and real data sets from the UCR time series repository. Furthermore, we explore the use of a multiple kernel learning framework for alignment-based classification of curves.

2. Infuse congealing with effective feature representations learned using unsupervised feature learning models such as convolutional restricted Boltzmann machines (Chapter 3). We show an improvement in alignment and classification performance on complex curve and face images due to the use of these feature representations.

3. Develop a nonparametric Bayesian joint alignment and clustering model that can simultaneously align and cluster a data set, where the number of clus-

ters is automatically discovered in a data-driven fashion (Chapter 4). This model shares the benefits of congealing (i.e. nonparametric, separation of transformation operator and optimization procedure), but can better handle complex/multi-modal data sets. While it is possible to perform each of the two processes (alignment and clustering) in sequence, we show that performing them simultaneously is more beneficial given their dependencies. For example, clustering is complicated by the fact that instances are out of alignment. We apply this model to both curve and image data sets showing improvements in alignment accuracy over congealing, and improvements in clustering accuracy over KMeans, infinite mixture models [18], and affinity propagation [20]. We also show large improvements on a challenging digits data sets to prior joint alignment and clustering work [59].

4. Develop a flexible and modular software library for alignment, clustering and feature learning (Chapter 5). Given the wide range of domains that benefit from these processes, delivering an extensible software package would be beneficial to the scientific community. This library will be released on GitHub to encourage updates and contributions. Associated with this software library is a collections of scripts that allow the replication of the results in this thesis.

Each of Chapters 2, 3, and 4 introduce technical concepts and prior work that is relevant to the content of that chapter. We conclude in Chapter 6 with an analysis of the thesis contributions and a discussion of possible directions for future work.

In summary, this thesis takes a positive step in improving the performance of joint alignment algorithms by utilizing unsupervised feature learning and nonparametric Bayesian clustering techniques in a sensible and effective manner. By testing these models on several curve and image data sets and releasing open source implementations we offer an **effective unsupervised data processing module that can align, cluster, and infer an appropriate feature representation**. We particu-

larly hope to impact fields outside of computer science such as radiology and biology where our models and code can be effective and adapt to different data sets with minimal input from the practitioner.

CHAPTER 2

CURVE ALIGNMENT USING CONGEALING

In this chapter we adapt the congealing framework to 1-dimensional curves [1]. We develop a parameterized set of nonlinear transformations that allow us to apply congealing to this type of data. The result is a flexible nonparametric curve alignment algorithm that makes limited assumptions about the data and accommodates a wide range of continuous transformations. We present positive results on aligning synthetic and real curve data sets and show how the byproducts of alignment (the aligned data and transformation parameters) can dramatically improve the performance of an SVM classifier. We conclude with a discussion on the two main drawbacks of congealing: choice of feature representation and inability to handle multiple modes which will motivate our two subsequent chapters. We begin, however, with an overview of the congealing framework.

2.1 Congealing Overview

Congealing iteratively optimizes a set of transformation parameters (associated with the data) using an information-theoretic objective function as a measure of joint alignment. Each data point (in our case a single curve, C_n) is associated with a transformation parameter, V_n. Congealing *simultaneously* searches over the space of all the data points' transformation parameters to find those that maximize a measure of *joint* alignment. More formally, when aligning N curves, congealing searches for the

[1] An earlier version of this work was published in the 2009 International Conference on Acoustics, Speech and Signal Processing [64].

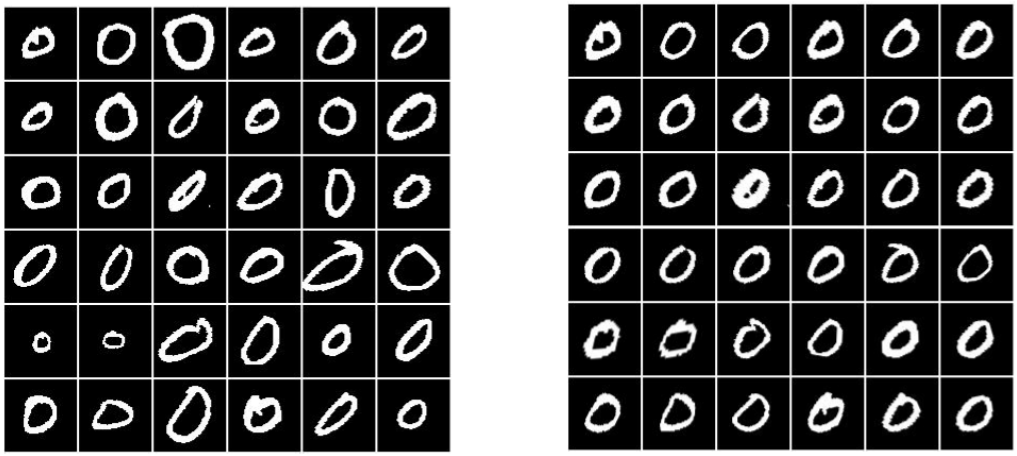

Figure 2.1: Alignment of binary images using congealing [66]. **Left:** before. **Right:** after.

transformation parameters $\{V_1, \ldots, V_N\}$ that maximizes a measure of the transformed curves, $\mathcal{S}(\mathcal{T}(C_1, V_1), \ldots, \mathcal{T}(C_N, V_N))$, where $\mathcal{T}(C_n, V_n)$ is the transformation function and $\mathcal{S}(C_1, \ldots, C_N)$ measures the joint alignment of a set of curves.

Congealing poses three key properties that make it a powerful and flexible:

- It performs joint alignment as opposed to pairwise alignment. While pairwise alignment has the advantage of performing a lower dimensional optimization, joint alignment algorithms have several advantages: (1) can utilize statistics of the entire data set, enabling them to avoid local minima, (2) do not rely on the manual selection of an appropriate reference instance which may not even exist, and (3) have the option to avoid the single modality assumption implicit in pairwise alignment algorithms.

- It is a data-driven approach since its measure of similarity (i.e. its objective function) is a measure of *joint* similarity of the (transformed) *data*.

- While a set of transformations needs to be specified, the framework itself places no restrictions on the transformations that are allowed (or their parameteriza-

Figure 2.2: Alignment of complex images using congealing [34]. **Left:** images of cars with a red bounding box representing the canonical location, orientation, and scale. **Right:** Similarly, for images of faces.

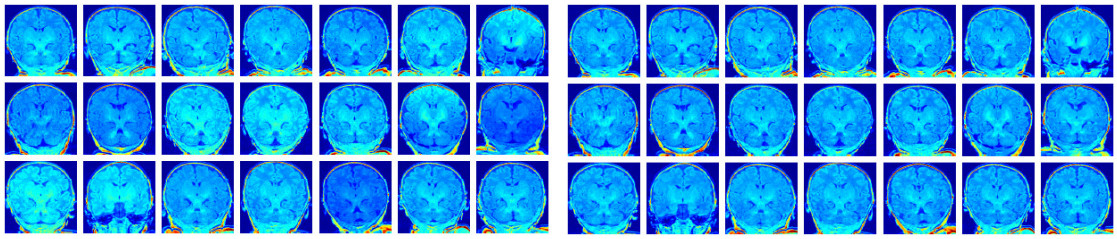

Figure 2.3: Bias removal result using congealing [50]. **Left:** before. **Right:** after.

tion). This allows congealing to be utilized to remove many forms of variability, including brightness [50].

The freedom to use any set of transformations, transformation parameterization, optimization procedure and similarity function makes congealing more of a framework than a specific algorithm. Different choices of these factors have resulted in effective algorithms for: binary images of digits [66, 93, 92], binary images of noisy Drosophila imaginal disc [2], grayscale image patches [92], grayscale images of faces [14], complex grayscale images of cars and faces [34, 37], bias from MRI images [50] and 3D brain volumes [103].

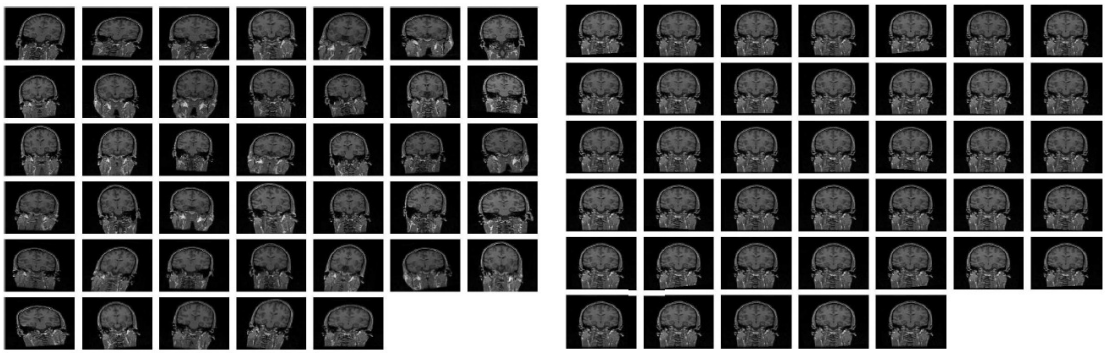

Figure 2.4: Alignment of 3D brain volumes using congealing [103]. **Left:** 2D slices before alignment. **Right:** 2D slices after alignment.

2.2 Algorithm

In this section we address each of the congealing design choices and present what we found to work well for joint alignment of 1-dimensional curves.

Allowable transformations. We allowed four transformations: nonlinear scaling in time, linear scaling in amplitude, nonlinear scaling in amplitude and amplitude translation. Nonlinear scaling implies that different regions of the curve are scaled by different amounts. Linear time scaling and time translation were not included since we assume that all the curves have the same length.[2]

Measure of joint alignment. We measured joint alignment using the sum of location-wise differential entropies. Each time step's entropy is independently calculated, treating the set of values from all curves at each time step as samples from a random variable with an unknown probability density function. Thus,

$$\mathcal{S}(C_1, \ldots, C_N) = -\sum_{k=1}^{K} \hat{H}_1(C_1(k), \ldots, C_N(k))$$

where N is the number of curves, K is the length of each curve, and $\hat{H}_1(\cdot)$ is a function that estimates the entropy of a unary random variable given samples from it. We used

[2] This is a common assumption for curve alignment and can easily be relaxed in our algorithm to allow translation and/or linear scaling in time.

the Vasicek [91, 3] estimator, which is a nonparametric spacings estimate of entropy that is the result of the uniformity of the *probability integral transform* of random variables and the re-substitution method for nonparametric entropy estimation:

$$\hat{H}_1(x^1, \ldots, x^N) \ = \frac{1}{N} \sum_{n=1}^{N-m_N} \ln \left(\frac{N}{m_N} \left(x^{(n+m_N)} - x^{(n)} \right) \right),$$

where $x^{(1)} \leq x^{(2)} \leq \cdots \leq x^{(N)}$ are the *order statistics* of the random sample, m_N is the order of the spacing (which depends on N, we set $m_N = \sqrt{N}$ in all our experiments), and $x^{(n+m_N)} - x^{(n)}$ is a *spacing of order* m_N.

Ideally we would compute the joint entropy of the curves by treating each one as a sample from an K-dimensional density. However, this estimation problem is infeasible because most curve data sets have more time steps than curves. Instead, the algorithm adds entropies at each time step, which corresponds to an implicit assumption that the time steps are independent after alignment. It is worth reminding the reader that,

$$H(X_1, \ldots, X_N) \leq \sum_{n=1}^{N} H(X_n)$$

where $H(\cdot)$ is the entropy function. Thus, by minimizing $\sum_{n=1}^{N} H(X_n)$, we are minimizing the upper bound on the joint entropy $H(X_1, \ldots, X_N)$.

We also experimented with a sum-of-variance objective function:

$$\mathcal{S}(C_1, \ldots, C_N) = -\sum_{k=1}^{K} \hat{\sigma}^2_{mle}(C_1(k), \ldots, C_N(k)),$$

where $\hat{\sigma}^2_{mle}(\cdot)$ is a function that computes the maximum likelihood estimate of the sample variance. Such a similarity measure would be more appropriate for data sets that are known to be unimodal, as our synthetic data sets in § 2.3.1.

Optimization procedure. We found that a variant of the conditional maximization procedure originally used to align binary digits [49] converged quickly and

avoided local minima. In each iteration this procedure iterates over all the transformation parameters for all the curves and updates their parameters by a *random* amount so as to increase the joint similarity. We found that randomizing (by sampling from a Uniform[0, Δ_p] - one for each transformation parameter, p) the transformation updates did a better job at avoiding local minima than picking pre-determined step sizes. At the end of each iteration we update the transformation parameters such that the mean transformation is zero. This helps regularize the parameters and ensures that large transformations are not inflicted on the data. Furthermore, the parameters of the Uniform distribution used to sample transformation parameters are dampened (by multiplying by a constant, λ, that is close to 1) to slowly reduce the transformation parameter changes with each iteration. Table 2.1 contains pseudo-code for the optimization procedure.

Transformation parameterization. One of the biggest difficulties in developing alignment algorithms is in parameterizing the transformations. Some parameterizations are simple, such as those for linear scaling and translations, but others, such as those for nonlinear time scaling, are more challenging. Therefore, several algorithms, including *dynamic time warping*, do not parameterize nonlinear scaling in time, and attempt to search all possible monotonic scaling functions. Other approaches exclude nonlinear scalings (e.g. [22]) or take a local approach to alignment (e.g. [58]) which can arbitrarily warp the curves. These approaches miss the benefits of good parameterizations, including low dimensionality, high modeling capacity and efficient computation.

Our algorithm allows changes in amplitude of the form $y(t) \approx \alpha g(t')x(t') + \beta$, where $x(t)$ is the original curve, $y(t)$ is the transformed curve, $t' = h(t)$ represents the monotonic (time) warping function, $g(t)$ represents the nonlinear amplitude warping function, α represents the linear amplitude scaling parameter, β represents the amplitude translation parameter, and the $\approx$ is due to bilinear interpolation. Transforma-

Algorithm: Curve alignment

Input: N curves: $C_1, \ldots, C_N$, transformation: $\mathcal{T}(C, V)$, similarity: $\mathcal{S}(C_1, \ldots, C_N)$
Parameter: maximum step size: $\Delta_p > 0$, step size scaling factor: $0 < \lambda \leq 1$
Output: N aligned curves: $C'_1, \ldots, C'_N$, N parameter vectors of length P: $V_1, \ldots, V_N$

Initialize parameter vectors, $\forall_{1 \leq n \leq N}\ V_n = \mathbf{0}^P$
Initialize transformed curves, $\forall_{1 \leq n \leq N}\ C'_n = C_n$
Initialize joint similarity, $H = \mathcal{S}(C'_1, \ldots, C'_N)$
Until *convergence* {
    for $n = 1$ to N { *iterate over curves*
        $\hat{V}_n = V_n$
        for $p = 1$ to P { *iterate over parameters*
            $\delta = rand(0, \Delta_p)$ *sample from* Unif$(0, \Delta_p)$
            $V_{tmp} = V_n$
            $V_{tmp}(p) = V_{tmp}(p) + \delta$ *update parameter*
            $C'_{tmp} = \mathcal{T}(C_n, V_{tmp})$ *transform curve*
            $H_{tmp} = \mathcal{S}(C'_{tmp}, C'_1, \ldots, C'_{n-1}, C'_{n+1}, \ldots, C'_N)$ *compute joint similarity*
            if $H_{tmp} > H$
                $\hat{V}_n(p) = V_{tmp}(p)$ *save new parameter*
            else { *try other direction*
                $V_{tmp}(p) = V_{tmp}(p) - 2\delta$
                $C'_{tmp} = \mathcal{T}(C_n, V_{tmp})$
                $H_{tmp} = \mathcal{S}(C'_{tmp}, C'_1, \ldots, C'_{n-1}, C'_{n+1}, \ldots, C'_N)$
                if $H_{tmp} > H$
                    $\hat{V}_n(p) = V_{tmp}(p)$
            }
        } *loop over parameters*
    } *loop over curves*
    $\Delta = \lambda\Delta$ *reduce maximum step size*
    $\forall_{1 \leq n \leq N}\ V_n = \hat{V}_n$ *retrieve saved parameters*
    $\forall_{1 \leq p \leq P}\ \mu(p) = mean(V_1(p), \ldots, V_N(p))$ *compute mean param*
    $\forall_{1 \leq n \leq N}\ V_n = V_n - \mu$ *set mean param to* 0
    $\forall_{1 \leq n \leq N}\ C'_n = \mathcal{T}(C_n, V_n)$ *transform curves*
    $H = \mathcal{S}(C'_1, \ldots, C'_N)$ *compute joint similarity*
} *convergence loop*
return $\{C'_1, \ldots, C'_N\}$ and $\{V_1, \ldots, V_N\}$

Table 2.1: Pseudo-code for our stochastic curve congealing algorithm.

tions are applied in the following order: nonlinear time warping, nonlinear amplitude warping, linear amplitude warping, and amplitude translation.

There are several ways to parameterize $h(t)$ (e.g. [87]). We found that Ramsay's method [76] using a Fourier basis (as opposed to the recommended b-splines) was the most efficient and compact — only 4 basis functions were needed. Ramsay's method is based on the realization that strictly monotonically increasing functions (with smooth first derivatives), $h(t)$, are a family of functions defined by the following differential equation,

$$\frac{\partial^2 h}{\partial t} = w \frac{\partial h}{\partial t}$$

where $w(t)$ is an unconstrained coefficient function. The above differential equation, subject to the constraints $h(0) = 0$ and $h(1) = 1$, has the following solution,

$$h(t) = \frac{1}{Z} \int_0^t \left[\exp\left(\int_0^t w(t)\, dt \right) \right] dt,$$

where Z is the normalizing constant,

$$Z = \int_0^1 \left[\exp\left(\int_0^1 w(t)\, dt \right) \right] dt.$$

This allows us to estimate an unconstrained coefficient function $w(t)$ and then use the above mapping to obtain the warping function, $h(t)$. $w(t)$ represents the relative curvature at each point, t. In the same way that we use $\exp(\cdot)$ to enforce positivity, the above mapping allows us to enforce monotonicity.

For $w(t)$ and $g(t)$, which are both unconstrained warping functions (the former for nonlinear time and the latter for nonlinear amplitude), we used a linear combination of sine and cosine functions at varying frequencies to parameterize the coefficient function. Thus,

$$w(t) = \sum_{k=1}^{|f|} \omega_k \sin(2\pi f_k t) + \omega_{k+|f|} \cos(2\pi f_k t)$$

where f is a vector of frequencies, and the weights $(\omega_1, \ldots, \omega_{2|f|})$, are the parameters for nonlinear time scaling. Similarly for $g(t)$, but with a different frequencies. We found that using only two frequencies, $(\frac{1}{2}, 1)$, for $w(t)$ and seven frequencies, $(\frac{1}{4}, \frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}, 3)$, for $g(t)$ provided sufficient modeling capacity.

Summary. These design choices resulted in an efficient and flexible algorithm for the joint alignment of curve data. It employs a search procedure that makes no assumptions about the form of the curves and only weak assumptions about the structure of transformations. Furthermore, each curve's transformation is parameterized by only 20 parameters (4 for nonlinear time, 14 for nonlinear amplitude, 1 for linear amplitude and 1 for amplitude translation).

2.3 Experimental Results

As initially highlighted in § 1.1, it is difficult to quantitatively evaluate the performance of alignment algorithms and qualitative evaluations are most meaningful if they are performed by an expert who understands the nature of the data set. In practice, the quality of alignment depends on the application. For example, measuring the sum of squared differences could be misleading because a collection of curves can always be trivially aligned — by setting them all to zero, for example — and even many non-trivial "perfect" alignments might be removing useful information. Therefore, we performed a series of alignment and classification experiments on synthetic and real data sets from the UCR time series repository [44] [3] to evaluate our algorithm. Using classification performance to measure alignment quality allows us to determine if the algorithm is providing a tangible benefit. It also allows us evaluate the benefits of joint alignment compared to pairwise alignment.

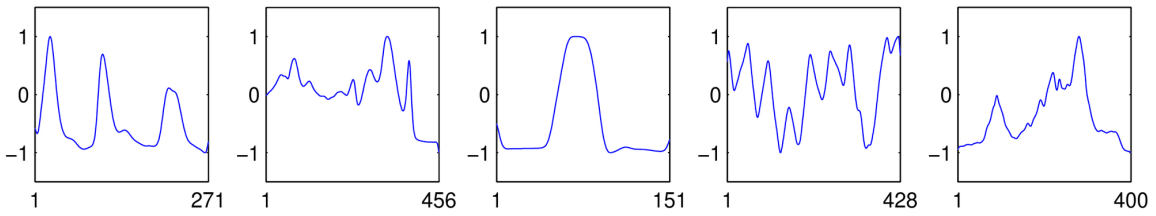

Figure 2.5: The five original curves used to generate the 75 synthetic data sets.

2.3.1 Evaluation using Synthetic Data Sets

The purpose of the first series of experiments was to evaluate the effectiveness of the algorithm's parameterization and search technique using synthetic data sets. We selected four curves from the UCR time series repository and 1 curve from the Total Ion Count data set [58] (see Figure 2.5). For each curve we created 15 synthetic data sets using 5 transformation groups and 3 levels of difficulty giving us a total of 75 data sets.[4] The 5 transformation groups included: only nonlinear time scaling, only linear amplitude scaling, only nonlinear amplitude scaling, only amplitude translation, and all four simultaneously. Each data set was created by randomly transforming the original curve 50 times. These data sets allow us to study each transformation independently, and provide us with the alignment ground-truth (the original curve used to generate the data set). We aligned each of the 75 data sets with our algorithm using the variance objective function where for each data set we only used that transformations that created it. This allowed us to more accurately assess our transformation parameterization given that the data sets are unimodal.

After alignment, we computed the average sum of squared (Euclidean) distance across all pairs of curves. A histogram of these scores across all 75 data sets is shown in Figure 2.10. Figures 2.6, 2.7, 2.8,and 2.9 show alignment results on data sets

[3] `http://www.cs.ucr.edu/~eamonn/time_series_data/`

[4] Note that these synthetic data sets are different from those used in our prior work [64]. Here, we tried 5 different transformation groups instead of just 3 and used 3 difficulty scales (Easy, Medium, and Hard) instead of 5. Overall, the data sets used in this thesis are more challenging.

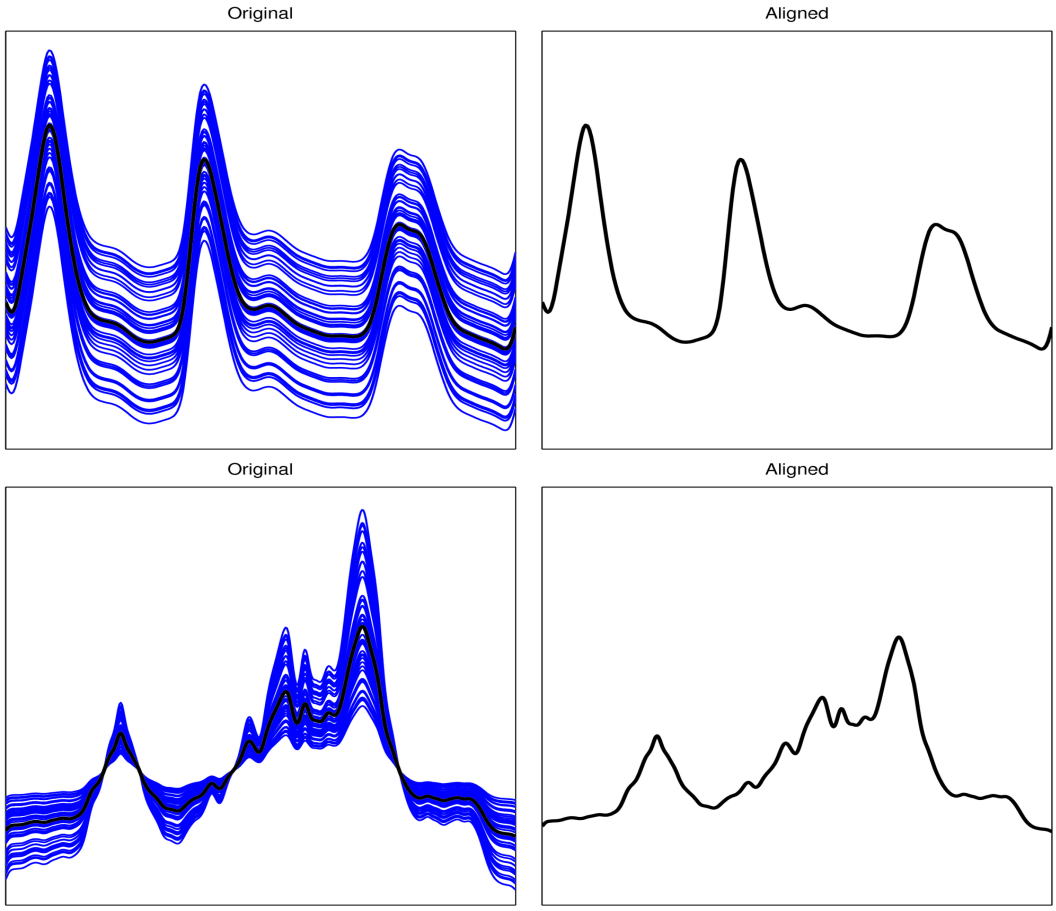

Figure 2.6: Alignment results on two synthetic data sets. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. **Top row:** Synthetic data set generated with amplitude translation. **Bottom row:** Synthetic data set generated with linear scaling. For both data sets, the curves were perfectly aligned and are hidden behind the mean curve.

with a difficulty level of 3 (most difficult). Overall, we found that for the majority of the data sets, our algorithm converged to a solution that is near the ground-truth. Specifically, data sets generated with linear amplitude scaling and amplitude translation were almost always perfectly aligned. There were a few data sets (less than 10%) generated with the three other transformation groups where the algorithm got trapped in local minima. Overall, this suggest that our algorithm has the capacity to align curve data corrupted by large transformations while avoiding local minima.

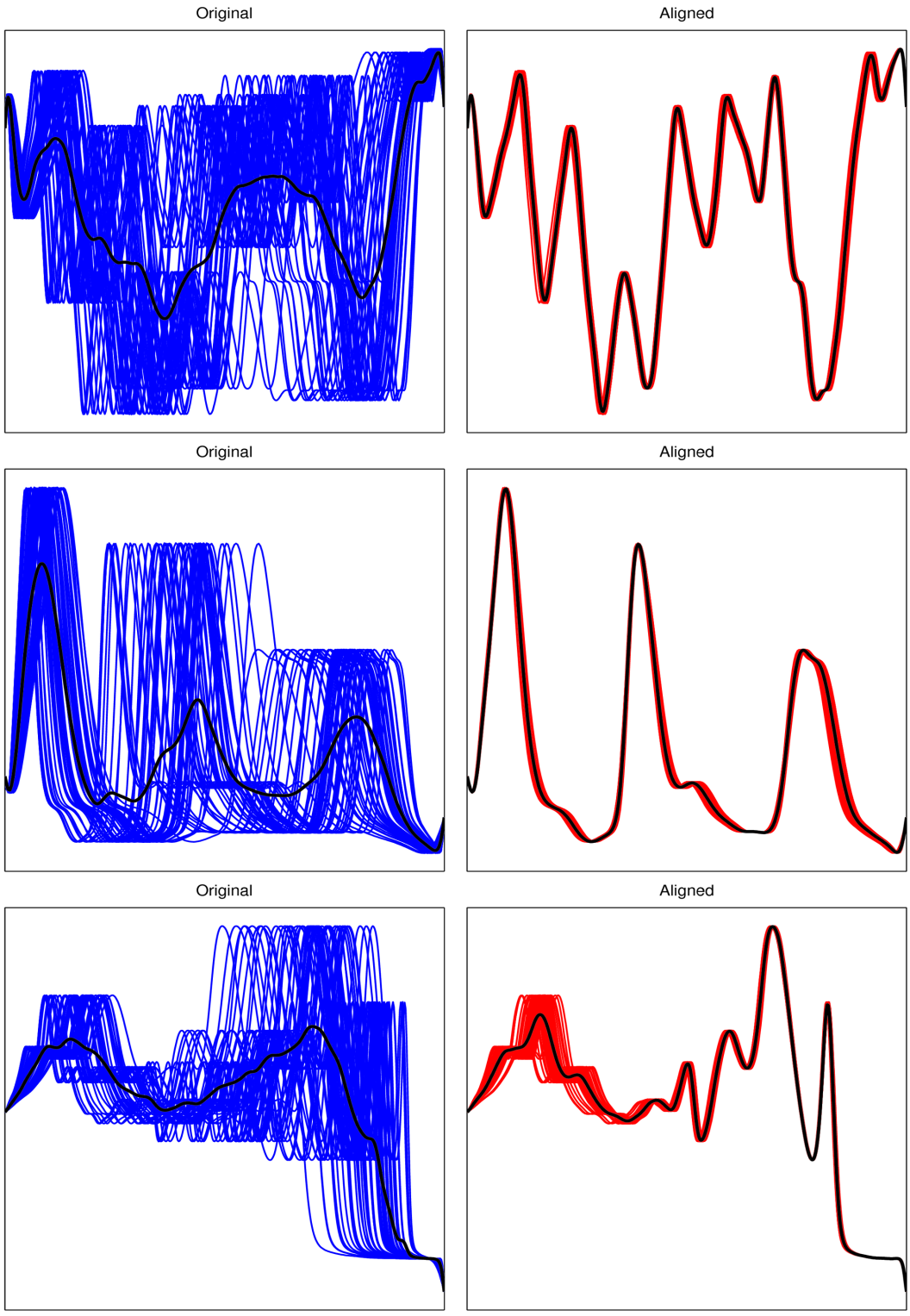

Figure 2.7: Alignment results on synthetic data sets generated with nonlinear time warping. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. The last row contains the worst alignment result for this transformation group.

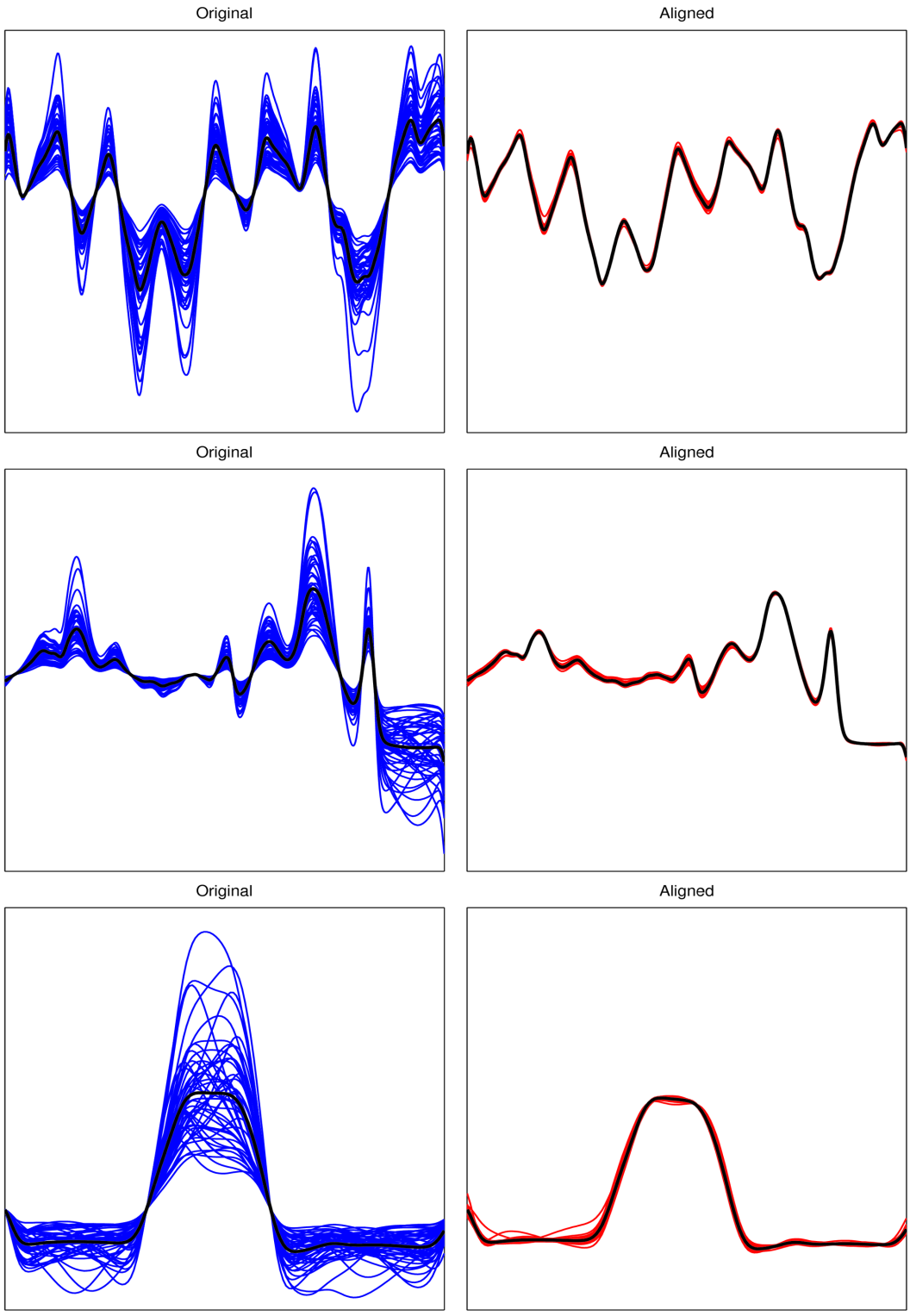

Figure 2.8: Alignment results on synthetic data sets generated with nonlinear amplitude warping. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. The last row contains the worst alignment result for this transformation group.

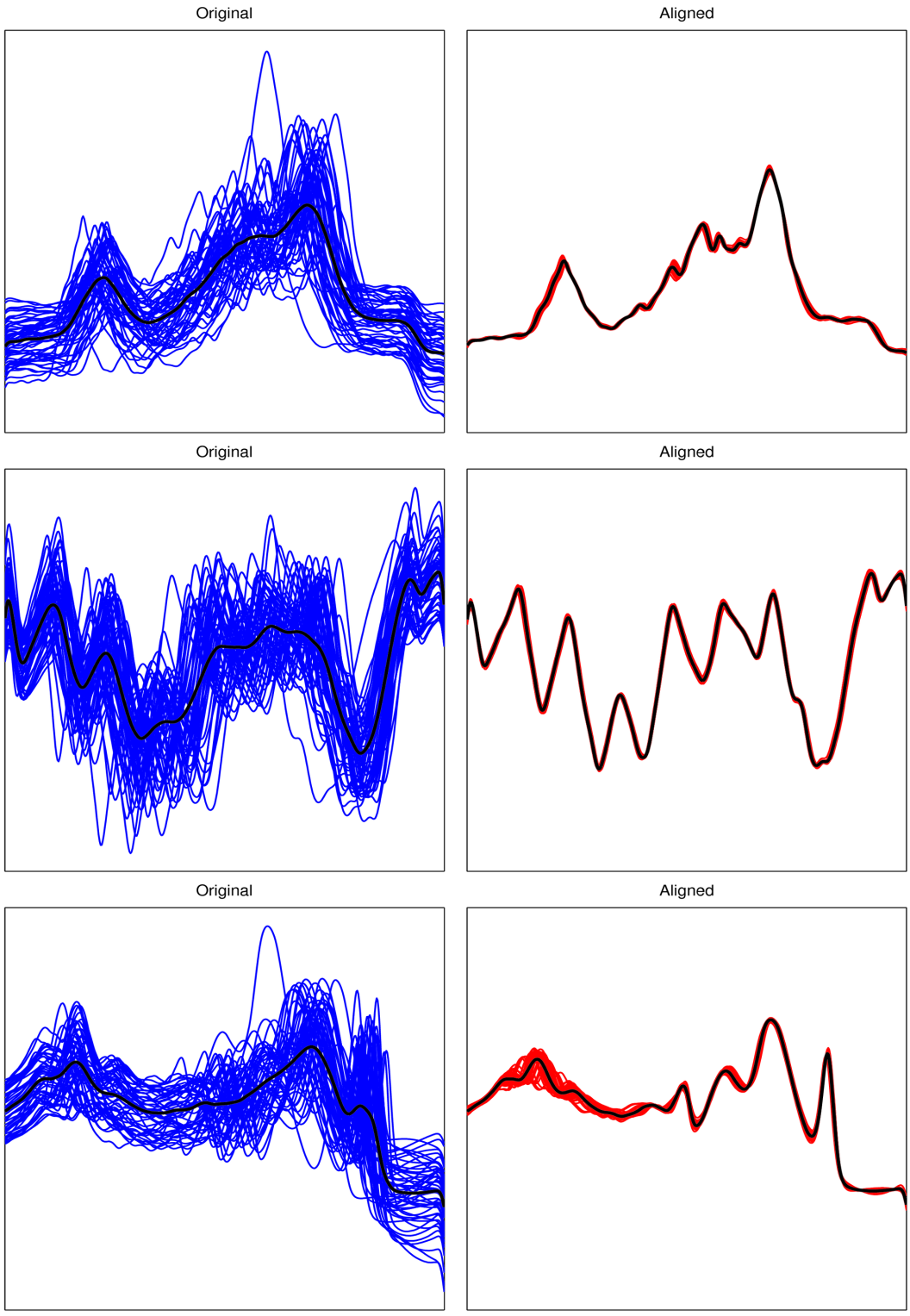

Figure 2.9: Alignment results on synthetic data sets generated with all four transformations. Original curves are in the first column (blue) and aligned curves are in the second column (red). The black curve in each plot is the mean curve. The last row contains the worst alignment result for this transformation group.

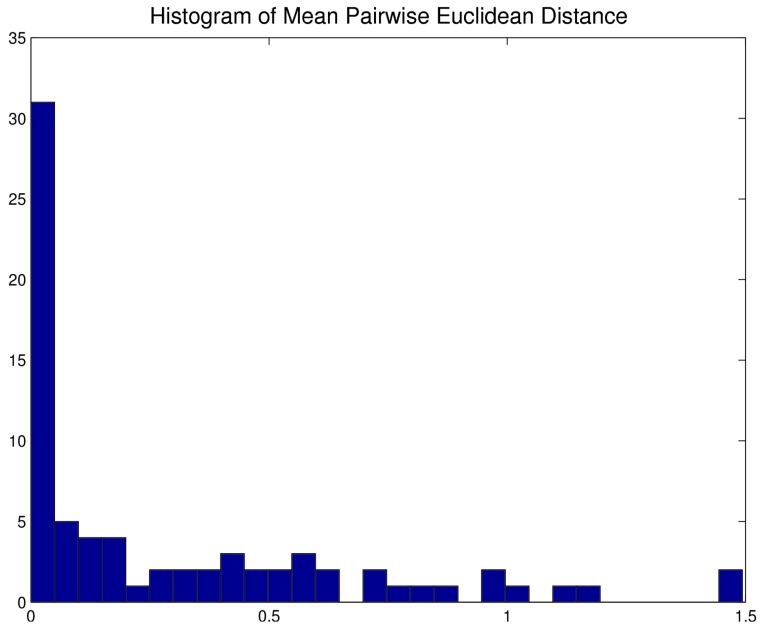

Figure 2.10: Histogram of the alignment scores across all 75 synthetic data sets. The alignment score is the average pairwise Euclidean distance between the aligned curves.

2.3.2 Evaluation using Real Data Sets

In this section we present results for our curve congealing algorithm on real data sets. We also compare the entropy and variance objective functions.

One of the benefits of using the entropy objective function is that it has a tendency to preserve modes in the data set, unlike variance which will guide the alignment so that all the aligned curves are most similar to each other. To highlight these differences we ran our algorithm using both objective functions on the Total Ion Count data used by CPM [58]. As Figure 2.11 illustrates, alignment with entropy did a better job at preserving the different "groups" in the data set (for example, the black and purple colored curves have amplitudes around the first bump that are different from the rest of the data set). Using the variance objective function, all the curves were aligned to each other compactly, generating an output similar to CPM despite having many fewer parameters and completing in a fraction of the running time. We also present alignment results on three additional data sets from the UCR

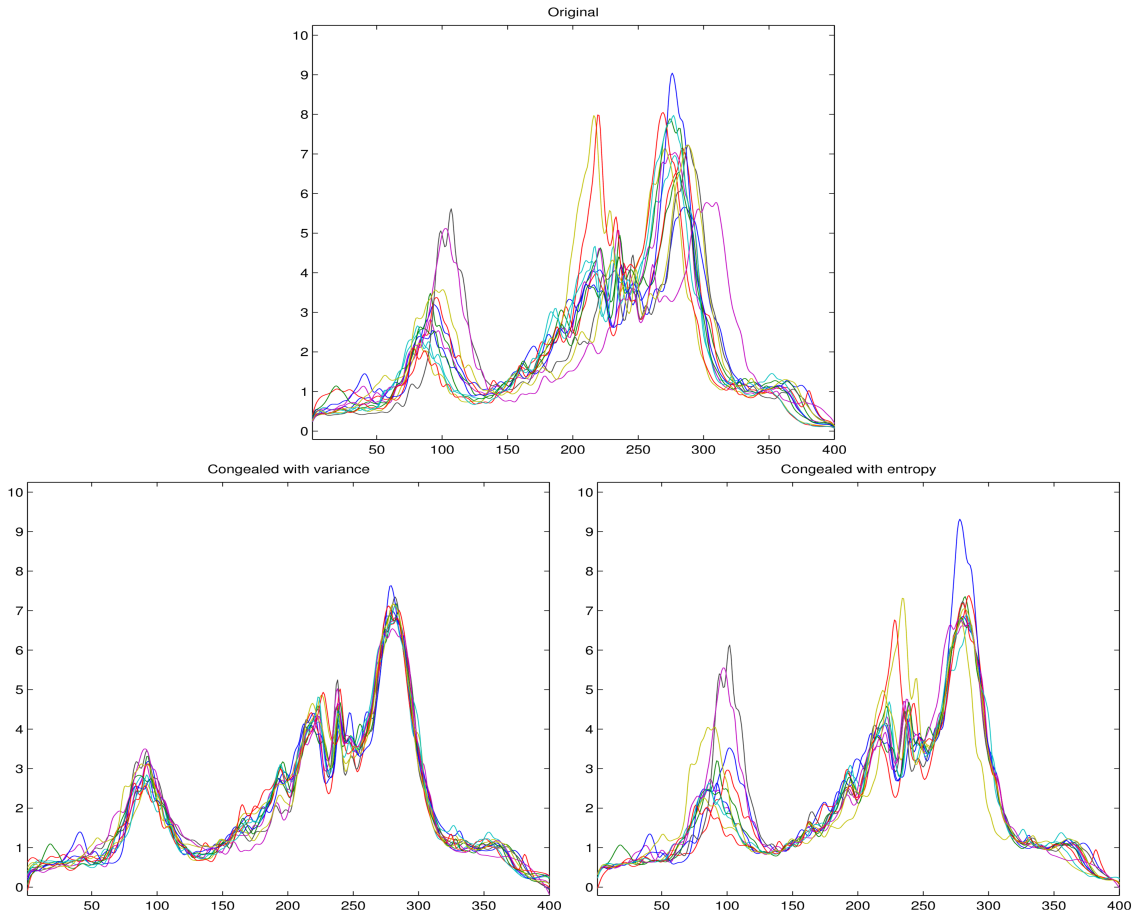

Figure 2.11: Alignment of the Total Ion Count data set [58] using congealing. **Top**: original. **Bottom-left**: aligned with variance. **Bottom-right**: aligned with entropy.

time series repository. Figure 2.12 shows an alignment (with entropy) on the ECG200 data set, Figures 2.13 and 2.14 shows alignments (with variance) on the Gun_Point and FaceFour data sets.

2.3.3 Improving Classification with Unsupervised Alignment

Given that the purpose of alignment is to eliminate undesirable variation, the performance of an alignment algorithm can be assessed by investigating whether it improves the performance of a classifier [34]. Often, bringing curves into correspondence simplifies the classification problem and improves a classifier's performance [99]. Classification based on unsupervised alignment involves aligning the train and test

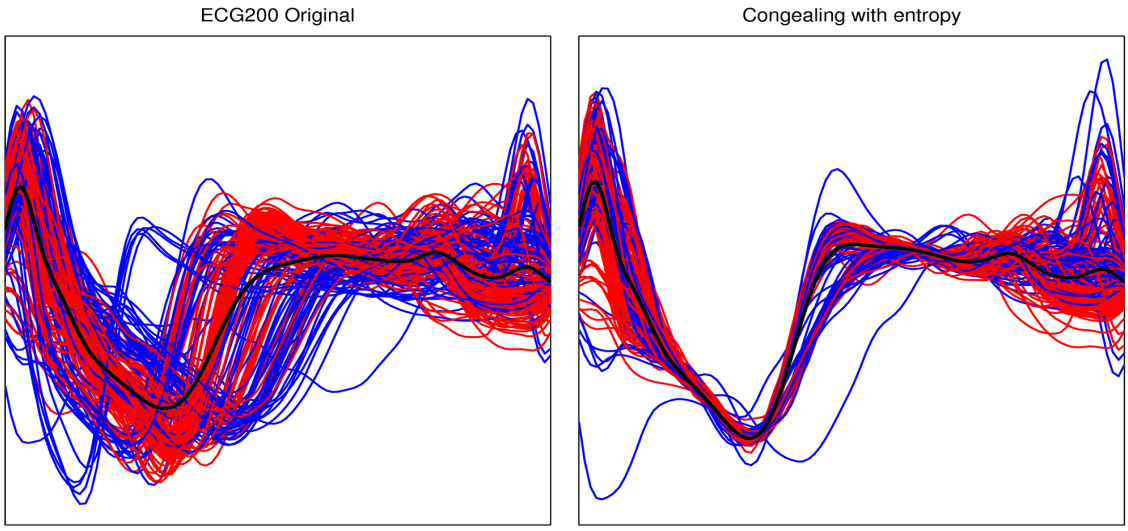

Figure 2.12: Alignment of the ECG200 data set (both classes) using congealing with entropy. **Left:** original curves before alignment, color-coded by class. **Right:** aligned curves, also color-coded by class. The black curve in each plot represents the mean curve.

curves from *all* the classes simultaneously, *without knowing the class label for each curve*, and then performing classification. We evaluated how alignment improves the performance of a linear support vector machine (SVM) classifier on thirteen data sets from the UCR time series repository (which have pre-defined train and test splits).

Alignment factors an original data set into an aligned data set and corresponding transformation parameters. We experimented with two methods for incorporating the aligned data set and transformation parameters into an SVM classifier:

1. Simple concatenation. In this set-up, we simply concatenated the original curve, the aligned curve and the transformation parameters into a single vector used to represent each curve. The length of this representation is $2 \cdot L + P$ where L is the length of the curve and P is the number of transformation parameters.

2. Multiple kernel learning [25]. In this set-up, we first learned separate (linear) kernels for each of the original data, aligned data, and transformation parameters and then combined these (base) kernels non-linearly [13].

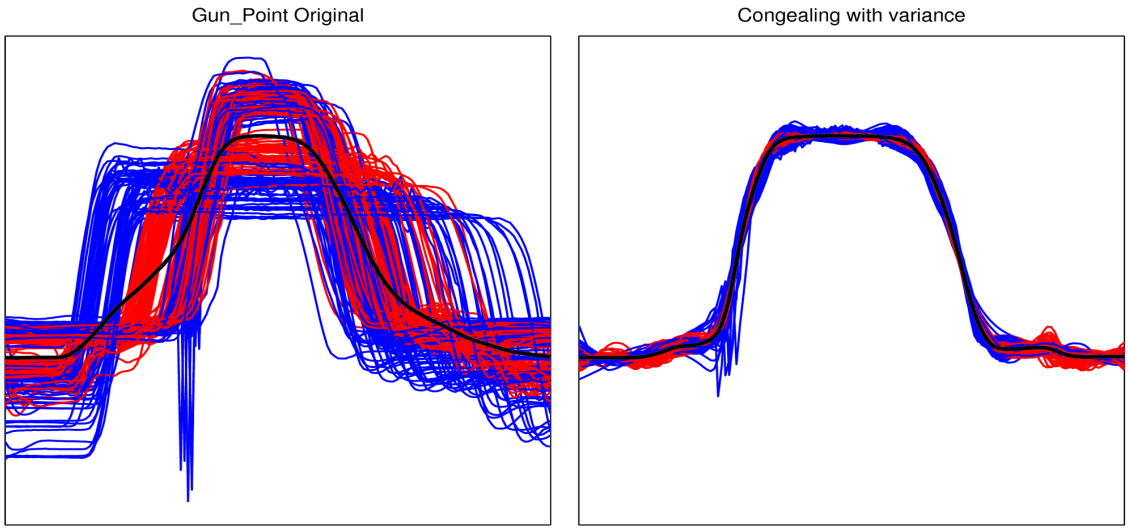

Figure 2.13: Alignment of the GunPoint data set (both classes) using congealing with variance. **Left:** original curves before alignment, color-coded by class. **Right:** aligned curves, also color-coded by class. The black curve in each plot represents the mean curve.

We also experimented with two transformation groups (nonlinear time scaling and all the transformations) to evaluate the benefit of utilizing all the transformations. When using all the transformations, we experimented with both the Vasicek entropy and variance objective functions. In addition to these 6 experimental settings, we also present results for two baselines: classification using the original curves alone, and classification using dynamic time warping. The former baseline allows us to evaluate the effect of our alignment algorithm on a linear SVM classifier, while the latter allows us to evaluate the benefit of joint alignment (our work) to pairwise alignment (dynamic time warping).

The next two subsections offer brief overviews of dynamic time warping (one of the baselines) and nonlinear combinations of kernels (one of the combination techniques). A reader familiar with these two topics may continue on to experimental results in § 2.3.3.3.

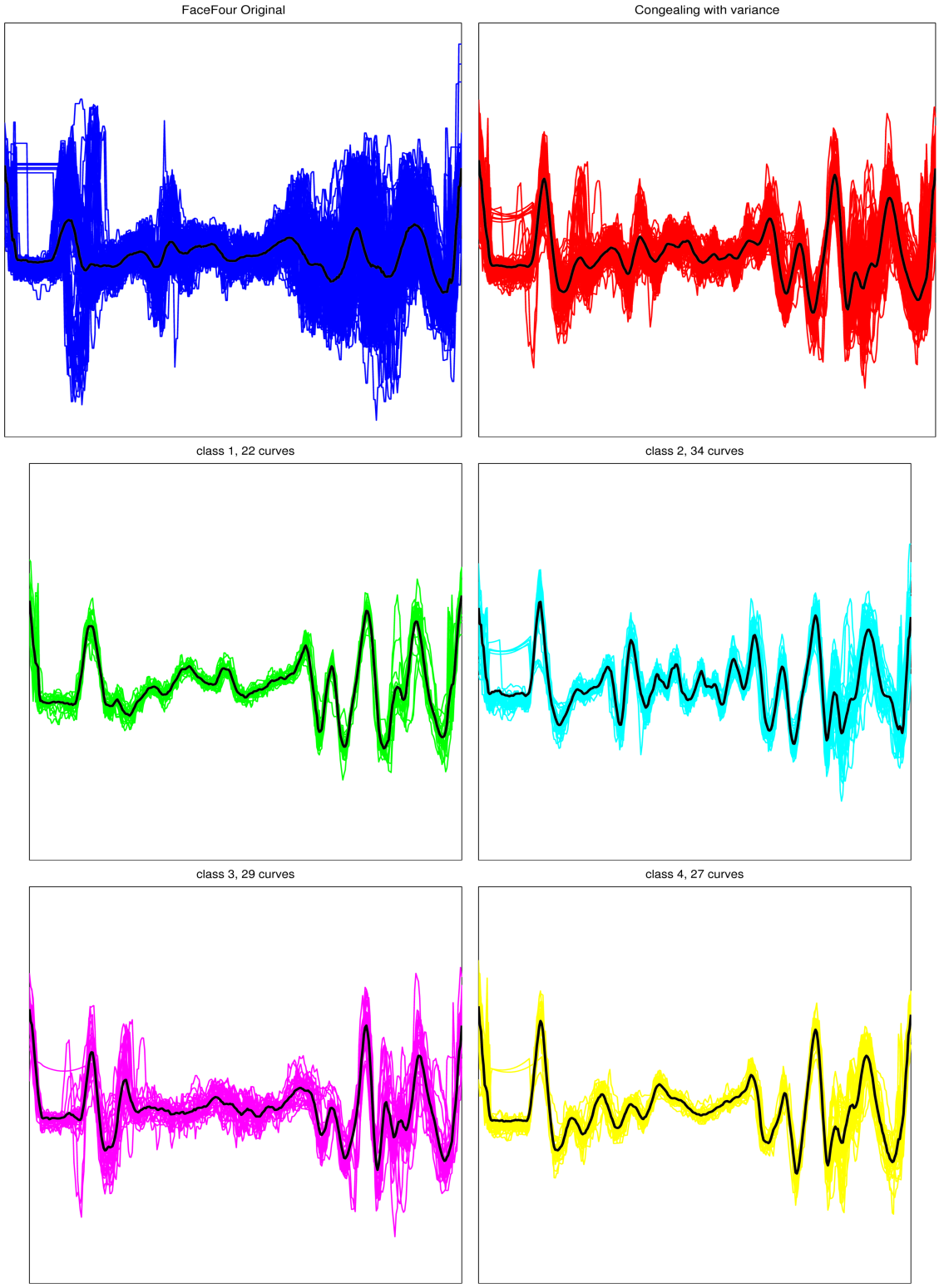

Figure 2.14: Alignment of the FaceFour data set (four classes) using congealing with variance. The **top row** displays the curves from all four classes before (left, blue) and after (right, red). The **middle** and **bottom** rows contain the curves in each of the four classes *after* alignment. The black curve in each plot represents the mean curve.

2.3.3.1 Interlude: Dynamic Time Warping

Dynamic time warping (DTW) is a procedure for aligning two sequences. It was initially used to align speech utterances [94], but has since been well studied [85, 69, 68, 80], adapted to a variety of other domains [1, 81, 99, 102] and scaled to handle large data sets [84, 75]. In many application areas, DTW has been shown to be very effective. For example, one of the most accurate techniques for curve classification is a nearest neighbor classifier using DTW distances [99].

DTW takes as input two sequences and finds a monotonic mapping between the elements of the sequences that minimizes their Euclidean distance. More formally, given two sequences $f(t)$ and $g(t)$, DTW searches for the best monotonic function $h(t)$ such that the Euclidean distance between $f(t)$ and $g(h(t))$ is minimum ($f(t)$ and $g(t)$ are exchangeable since there is no ordering on the input sequences). This can be solved using dynamic programming.

Consider the distance matrix D, where element (i, j) is $(f(i) - g(j))^2$. Any monotonic path in this matrix starting at the bottom-left $(1, 1)$ cell and ending at the top-right (N, N) cell (assuming both $f(t)$ and $g(t)$ are of length N, although this is not necessary) is a valid mapping between the two sequences and the sum of all the cell entries along this path represents the Euclidean distance between the two sequences under this mapping. DTW searches for the monotonic path through this matrix that has minimum total weight. This can be solved with dynamic programming using the following recurrence relation:

$$E(i, j) = min\{E(i-1, j), E(i, j-1), E(i-1, j-1)\} + D(i, j)$$

where $E(N, N)$ is the minimum Euclidean distance.

Solving the above recurrence relation computes an unconstrained warping function (represented by the monotonic path through the matrix) that maps the two sequences to each other. It is straightforward to add constraints to the warping function by

limiting the search window within the distance matrix. Such constraints play the role of regularization that prevents excessive warping which may be undesirable for some data sets. Using such constraints can significantly improve the performance of a nearest neighbor classifier with DTW distances on a variety of curve data sets. [5]

2.3.3.2 Interlude: Multiple Kernel Learning

Kernel methods for both supervised and unsupervised learning rely on the existence of a valid (positive definite) kernel function between two instances [32]. The choice of a kernel function is imperative to the success of the learning algorithm. Ultimately, the choice of a kernel function is left up to the user, which can be a daunting task that involves a lot of experimentation. A more appealing alternative is for the user to provide a set or family of possible kernel functions (typically called base kernels) and a labelled data set. Then the choice of the kernel function can be determined automatically, either by selecting the most appropriate function from the set or learning a new kernel that is a linear or non-linear combination of the base kernels. A variety of algorithms exist for kernel learning, many of which are nicely summarized by Gonen and Ethem [25].

Multiple kernel learning can also be used as a principled framework for combining different sources of information, such as global and local image features [57]. The process of alignment described in this chapter factors an input data set into an aligned version alongside transformation parameters. In our classification setting, we experimented with treating each of these three views as independent sources of information and used multiple kernel learning to combine them. The main advantage of this set-up is that it allows use to potentially utilize different kernel functions for each of these three sources (we expand on this in Chapter 6) and then combine them in a

[5] `http://www.cs.ucr.edu/~eamonn/time_series_data/`

principled manner. For all the experiments in this chapter, we utilized the non-linear combination method described by Cortes *et al.* [13].

2.3.3.3 Experimental Results

Table 2.2 presents the classification performance of our 6 experimental settings as well as the two baselines. Figure 2.15 contains two scatter plots that compare the classification performance of our best alignment result to both baselines. Examining these results enable us to perform analyses across different dimensions:

1	2	3	4	5	6	7	8	9
Alignment	None	DTW	Joint (with Congealing)					
Objective			Entropy		Var.	Entropy		Var.
Trans.			Time	All		Time	All	
Grouping			Concatenation			Multiple Kernel Learning		
Coffee	96.43	82.10	92.86	**100.00**	**100.00**	89.29	**100.00**	**100.00**
Beef	**83.33**	50.00	73.33	70.00	80.00	70.00	60.00	80.00
OliveOil	86.67	**86.70**	86.67	86.67	86.67	83.33	86.67	86.67
FaceFour	88.64	83.00	93.18	**97.73**	93.18	90.91	97.73	93.18
Lighting2	68.85	**86.90**	70.49	72.13	68.85	75.41	75.41	72.13
Lighting7	69.86	**72.70**	68.49	65.75	69.86	65.75	67.12	71.23
ECG200	81.00	77.00	79.00	83.00	84.00	**85.00**	**85.00**	82.00
Trace	82.00	**100.00**	91.00	89.00	92.00	90.00	89.00	93.00
Gun_Point	90.67	90.70	92.00	93.33	**99.33**	94.67	96.67	98.00
FISH	85.14	83.30	93.14	91.43	94.29	**94.86**	90.86	**94.86**
OSULeaf	43.39	**59.10**	46.28	41.74	50.83	53.31	50.00	54.13
synth. ctrl.	92.33	**99.30**	94.00	98.00	96.67	95.33	97.00	95.67
CBF	87.78	**99.70**	94.00	98.78	97.22	93.56	98.67	94.44
Mean	81.24	82.35	82.65	83.66	85.61	83.19	84.16	**85.79**

Table 2.2: Curve classification using a linear SVM. The first column contains the name of the UCR data set (one per row) and the remaining eight columns contain the classification performance using different techniques. Column 2: direct classification using a linear SVM classifier, without alignment. Column 3: classification using dynamic time warping (DTW). Columns 4 - 9: classification based on joint alignment. We experimented with 6 different settings which are indicated by the three rows labeled "Objective" (entropy vs. variance), "Trans." (all transformations vs. nonlinear time scaling), and "Grouping" (simple concatenation vs. multiple kernel learning).

- **Without alignment vs. With Joint Alignment.** Unsupervised joint alignment always improves the performance of a linear SVM, on average. The improvements of the different experimental settings range from +1.41% (column 4) to +4.55% (column 9), averaged across the 13 data sets.

- **Single transformation vs. All transformations.** Adding more transformations is beneficial. Aligning with all the transformation groups (nonlinear time scaling, nonlinear amplitude scaling, linear amplitude scaling, and amplitude) performs better, on average, than aligning with only nonlinear time scaling for both the concatenation and MKL groupings (column 5 vs. column 4, and column 8 vs. column 7).

- **Entropy vs. Variance.** Variance performs better than entropy when aligning with all the transformation groups for both the concatenation and MKL settings (column 6 vs. column 5, and column 9 vs column 8).

- **Joint Alignment vs. Pairwise alignment.** Joint alignment (columns 4 - 9) always performs better, on average, than pairwise alignment with DTW (column 3). The best joint alignment result (column 9) improves upon DTW (column 2) more than DTW improves upon the baseline (column 1), see Figure 2.15 (right). The reason behind this is that through joint alignment we can utilize both the aligned data and transformation parameters. For data sets where alignment removes the discriminability of the classes, our techniques have the advantage of being able to incorporate the transformation parameters which are crucial for these data sets. Furthermore, joint alignment provides a natural mechanism for regularizing the transformations since they utilize the statistics of the *entire* data set. Pairwise alignment techniques, such as DTW, typically miss out on such a global view.

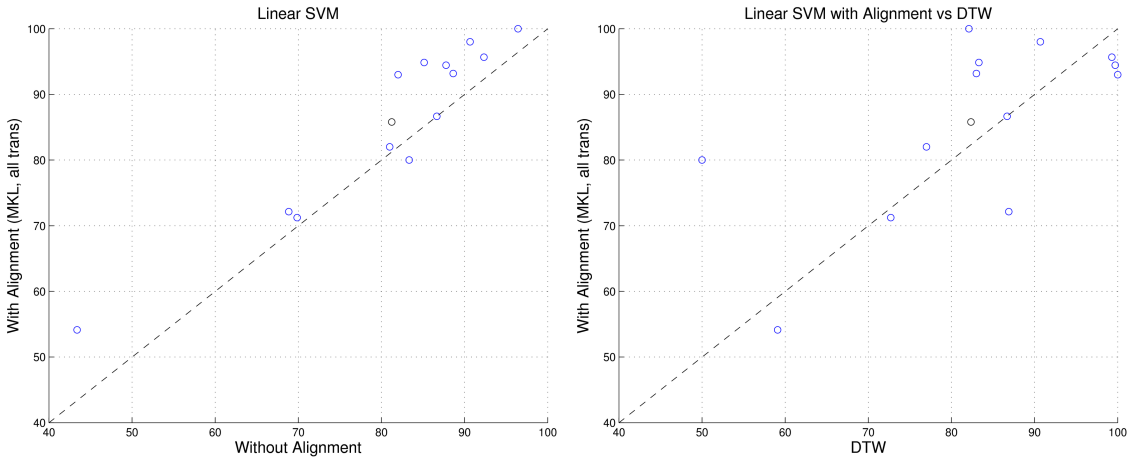

Figure 2.15: Classification scatter plots comparing two techniques. **Left:** Compares our best result (all transformations + variance + MKL) to the linear SVM baseline (no alignment). **Right:** Compares our best result (all transformations + variance + MKL) to dynamic time warping (DTW).

- **Single kernel vs. Multiple kernels.** Concatenation works surprisingly well, and MKL adds a moderate improvement. A positive characteristic of the best MKL setting (column 9) is that the classification accuracy on 12 of the 13 data sets was always at least as high as with just the original data (column 2), see Figure 2.15 (left). Many of the improvements are significant. For example on 6 of the 13 data sets, the improvement was $4.5\% - 11\%$. One key advantage of a MKL framework not utilized here is the ability to learn a kernel specific to the transformation parameters as opposed to a generic kernel such as a polynomial or radial basis function. Typically incorporating kernels that are tuned to the specific transformation parameters can outperform such generic kernels [65].

Overall these results not only support and highlight the effectiveness of our alignment algorithm, but also emphasize the utility of joint alignment as a mechanism for generating alternative views of a data set that can aid in classification. It is important to state that the same set of parameters for congealing was used for all thirteen data sets and no data set-specific changes or tuning was performed. This showcases the

1	2	3	4	5	6
Alignment	None	DTW	Joint (with Congealing)		
Objective			Entropy		Variance
Transformations			Time	All	
Grouping			Concatenation		
Coffee	89.29	82.10	92.86	**100.00**	**100.00**
Beef	80.00	50.00	70.00	66.67	**83.33**
OliveOil	83.33	**86.70**	83.33	86.67	86.67
FaceFour	85.23	83.00	90.91	**97.73**	93.18
Lighting2	75.41	**86.90**	75.41	73.77	73.77
Lighting7	71.23	**72.70**	64.38	67.12	71.23
ECG200	**86.00**	77.00	84.00	85.00	**86.00**
Trace	82.00	**100.00**	91.00	89.00	93.00
Gun_Point	94.00	90.70	94.67	96.00	**99.33**
FISH	85.71	83.30	94.29	90.86	**95.43**
OSULeaf	56.61	**59.10**	54.13	52.07	54.13
synth. ctrl.	95.67	**99.30**	95.00	97.33	96.00
CBF	87.67	**99.70**	93.33	99.00	97.56
Mean	82.47	82.35	83.33	84.71	**86.89**

Table 2.3: Curve classification using a second-degree polynomial SVM. The first column contains the name of the UCR data set (one per row) and the remaining five columns contain the classification performance using different techniques. Column 2: direct classification using a second-degree polynomial SVM classifier, without alignment. Column 3: classification using dynamic time warping (here, we include results reported on the UCR time series repository webpage). Columns 4 - 6: classification based on joint alignment. We experimented with 3 different settings which are indicated by the three rows labeled "Objective" (entropy vs. variance), "Trans." (all transformations vs nonlinear time scaling), and "Grouping" (simple concatenation).

generality of our alignment algorithm and its applicability to a wide range of eclectic data sets out-of-the-box.

We conclude our experimental section by exploring the effect on classification performance by moving from a simple linear kernel to a second-degree polynomial kernel. Specifically we investigate if the advantages provided by joint alignment for the linear kernel carry over to more complex kernels. Table 2.3 and Figure 2.16 present classification results using a second-degree polynomial kernel settings similar to the linear kernel. What these results highlight is that the trend we noticed for a linear

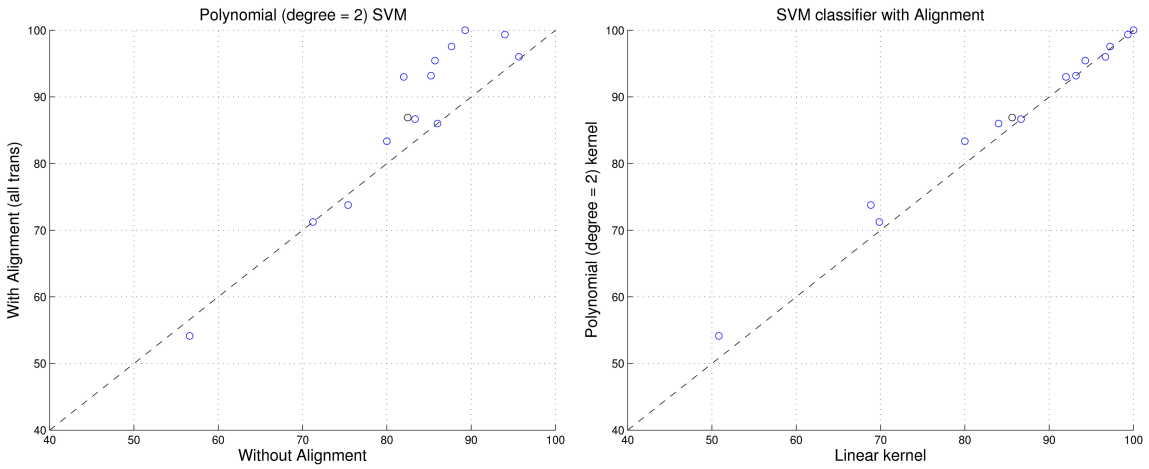

Figure 2.16: Classification scatter plots comparing two techniques. **Left:** Compares our best polynomial SVM result (all transformations + variance) to the polynomial SVM baseline (no alignment). **Right:** Compares our best polynomial SVM result (all transformations + variance) to our best linear SVM concatenation result (all transformation + variance).

kernel persists for a second-degree polynomial kernel, namely that joint alignment is beneficial, adding more transformations improves performance, and alignment with variance outperforms alignment with entropy.

2.4 Discussion and Drawbacks

We presented an efficient joint alignment algorithm for curve data, which demonstrated the utility of an efficient parameterization for nonlinear transformations. We tested our algorithm on a wide range of complex curve data sets and in almost all the cases it improved the correspondence across the curves and improved the performance of a classifier on those data sets. Compared to existing algorithms, it makes fewer assumptions about the distribution of the curves and the transformations, making it more widely applicable.

We'd like to highlight two key issues observed with our algorithm and the congealing framework in general. This will will serve as motivation for our subsequent

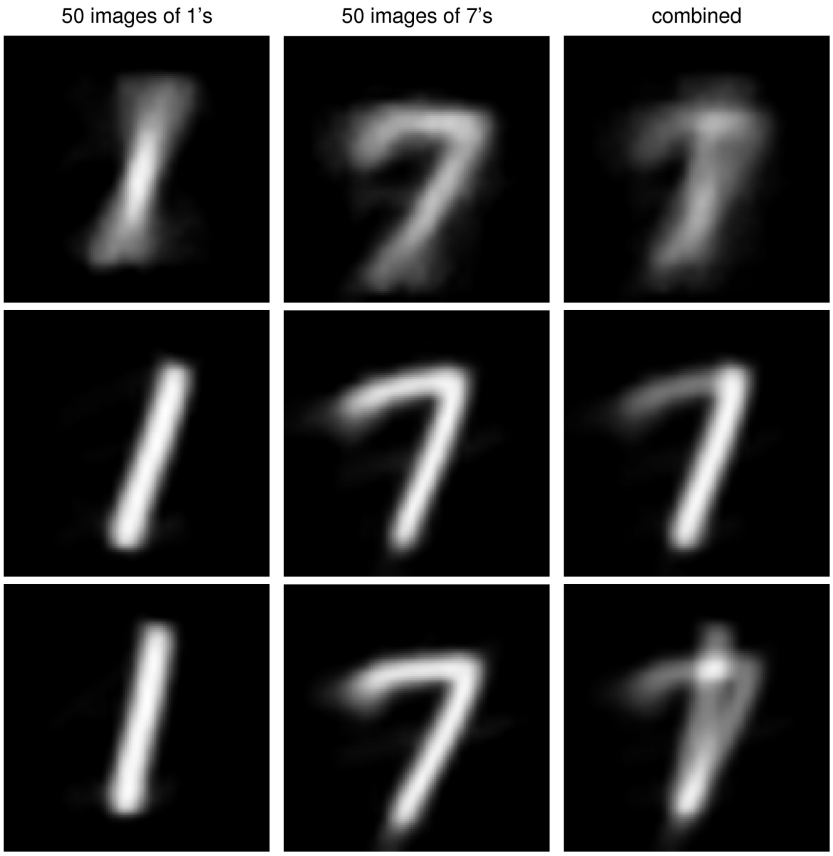

Figure 2.17: Effect of congealing a multi-modal data set of 1's and 7's. The **first row** contains the mean images of the original data set, the **second row** contains the mean images after congealing the entire data set, and the **third row** shows the mean images after congealing each of the digit classes separately.

work in Chapters 3 and 4. The two issues are that of feature representation and co-alignment due to the independence assumption in the objective function:

- **Feature representation.** In this chapter we only utilized the raw representation of curves and did not compute any higher-order features. The issue of representation and feature selection arises with each new data set and mode (e.g. faces, cars, and curves). Consequently it would be convenient if a representation could be automatically learned using the statistics of the data set.

- **Co-alignment.** One of the advantages highlighted for the entropy objective function in Figure 2.11 is its ability to preserve modes. However, an issue arises due to the independence assumption (temporally for curves and spatially for images) that results in *co-alignment*. Co-alignment occurs when two different groups (or clusters) of instances are partially aligned to each other. For example in Figure 2.11 if the black and purple curves are in fact a separate cluster, then it would be preferable if these two curves were aligned to each other without influence from the rest of the data set. Such a behavior might be achieved if we utilized a full joint density estimate for our entropy computation, but our independence assumption forces the middle segment of the curves to be aligned to each other despite other parts of the curves kept separate. To better understand the effect of co-alignment, consider congealing a data set of binary images containing the handwritten digits '1' and '7' (Figure 2.17). Notice that when the two digit classes are congealed together, the images of the digit '7' are transformed to align with the images of the digit '1'.

In the subsequent two chapters we address each of these issues. More specifically, in Chapter 3, we experiment with convolutional restricted Boltzmann machines [54] as a framework for unsupervised feature learning for improving congealing. We show improvements in congealing both curves and complex images of faces using the *exact* same feature learning framework. In Chapter 4, we consider the task of joint alignment and clustering as a mechanism for explicitly grouping items and avoiding the the bias introduced due to the independence assumption. The use of nonparametric Bayesian priors will allow us to maintain a nonparametric approach and discover the number of groups in a data-driven fashion. We show improvements in alignment *and* clustering quality over alignment-only (e.g. congealing), clustering-only (e.g. KMeans, affinity propagation [20], infinite mixture models [18]), and previous simultaneous alignment and clustering algorithms [21, 59].

CHAPTER 3

INCORPORATING FEATURE LEARNING

In this chapter we improve upon the congealing framework by using a high-order representation of the data to guide alignment. Specifically, when aligning (and classifying) the curves in the previous chapter, only the raw representation of the data was considered. We explore the benefits of incorporating a more robust feature representation and evaluate how it improves alignment and classification performance. We provide experimental results highlighting the utility of the convolutional restricted Boltzmann machine (CRBM) representation for the alignment of both curve and image data sets (separately). [1]

This issue of feature selection arises with every new data set. For example, in the subsequent chapter on alignment and clustering (Chapter 4), we had to alter the feature representation as we moved from a two-class digit data set of '4' and '9' to the full 10-class digit data set (specifically, moving from a raw pixel representation to a histogram of oriented gradients [15] representation). Furthermore, we found this histogram of oriented gradient representation ineffective on a data set of face images from the LFW database [36, 38] requiring us to experiment with alternate feature representations. Similarly, when extending the congealing algorithm from binary images of digits to complex images of cars and faces [34], the raw pixel representation was no longer sufficient and a higher-order representation based on clustering SIFT [61] features was needed for some success.

---

[1] Using CRBM's for improving congealing on images of faces was first published in the 2012 Neural Information Processing Systems conference [37] and later expanded [33]. This thesis focuses on using CRBMs for curve data following our work in Chapter 2.

In theory, there are many viable feature representations such as Fourier basis, wavelets, and principal components analysis. In this thesis we explore the use of features automatically learned using unsupervised feature learning algorithms. Three reasons guided this decision:

1. Many unsupervised feature learning frameworks are applicable to data sets of any dimensionality, such as 1-dimensional curves, 2-dimensional images, and 3-dimensional volumes.

2. They have received a resurge of interest from the machine learning community and have been effective in a variety of tasks including computer vision tasks [101, 54, 100, 40, 35], audio recognition [55], natural language processing [11], and information retrieval [83].

3. Since the issue of feature selection arises with every new data set, it would be convenient and effective if a representation could be *learned* automatically using the statistics of the points in the data set. Unsupervised feature learning is an effective framework for doing so.

We specifically experimented with convolutional restricted Boltzmann machines (CRBM) [2] given their positive performance on image classification tasks [54]. Furthermore, in a CRBM model, filters are convolved with the input which provides features that are temporally (or spatially) anchored which is beneficial for alignment. In the next section we briefly overview restricted Boltzmann machines (RBM) and CRBMs. For a full treatment on unsupervised feature learning and deep architectures we refer the reader to several great tutorials [4, 19] and theses [10].

[2] In this thesis, when referring to a convolutional RBM (CRBM) we are referring to the max-pooling CRBM model [54].

3.1 Background: Convolutional RBM

A CRBM is a variant of an RBM which in turn is a special kind of Boltzmann machine (BM). A BM, which can be regarded as a Markov random field, is an undirected network of random variables that can be observed (visible) or unobserved (hidden). An RBM adds the restriction that the graph is bipartite with pairwise connections only between a hidden node and a visible node (i.e. no visible-visible or hidden-hidden connections). Furthermore, an RBM typically contains dense connections between the visible and hidden layers (see Figure 3.1). The hidden units are typically binary and the visible units can be either binary of Gaussian. The energy function for an RBM with N binary hidden units and M Gaussian visible units is:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{n=1}^{N}\sum_{m=1}^{M} w_{nm} h_n \frac{v_m}{\sigma_m} - \sum_{n=1}^{N} b_n h_n + \sum_{m=1}^{M} \frac{(v_m - c_m)^2}{2\sigma_m^2}.$$

Each visible unit has a Gaussian noise standard deviation, σ_m, and a bias, c_m. Each hidden unit also has a bias, b_n. The pairwise connections between the hidden and visible layers are represented by an $N \times M$ weight matrix W.

In the CRBM, rather than having a single hidden layer, we have multiple hidden layers organized into groups, with a key difference (compared to an RBM): rather than fully connecting the hidden layer and visible layer, the weights between the hidden units and visible units are *local* and *shared* among all the hidden units in each group (see Figure 3.1). The CRBM captures the intuition that if a certain feature (or pattern) is useful in some locations, then the same feature can also be useful in other locations. The CRBM has three sets of parameters: convolution filter weights between the hidden nodes and the visible nodes (one per group); hidden biases $b^k \in \Re$ that are shared among hidden nodes (one per group); and visible bias $c \in \Re$ that is shared among all the visible nodes.

To make CRBMs more scalable, Lee *et al.* [54] developed *probabilistic max-pooling*, a technique for incorporating local translation invariance while allowing probabilistic

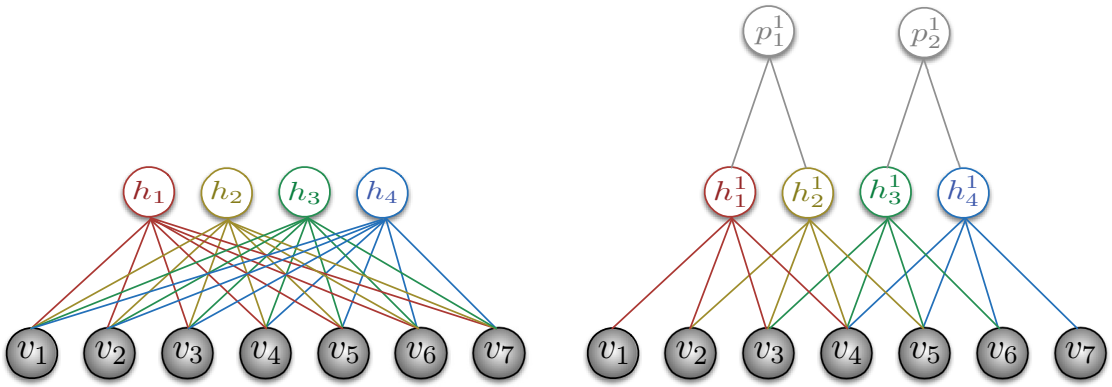

Figure 3.1: Boltzmann machines. **Left**: Standard RBM with 7 visible units and 4 hidden units. The graph is bipartite with dense (undirected) connections from every visible unit to every hidden unit. Each hidden unit is given a unique color that matches the color of its connections to the visible layer. Each edge between a hidden unit, h_i, and visible unit, v_j, has its own weight, w_{ij}. In this example, the weight matrix, W, is 4×7. **Right**: Max-pooling convolutional RBM also with 7 visible units and a single group (1 hidden layer consisting of 4 hidden units, and 2 pooling units). The superscript 1 for both the hidden and pooling nodes is to make explicit that they belong to the first group (the only group in this example). Again, each hidden unit is given a unique color that matches the color of its connections to the visible layer. However, unlike an RBM, each hidden unit is connected to contiguous (local) subset of the visible units and the weights of those connections are shared across all the hidden units. In this example, the following sets of connections share the same weight (the superscript is removed to make the notation less cumbersome): $\{(h_1, v_1), (h_2, v_2), (h_3, v_3), (h_4, v_4)\}$, $\{(h_1, v_2), (h_2, v_3), (h_3, v_4), (h_4, v_5)\}$, $\{(h_1, v_3), (h_2, v_4), (h_3, v_5), (h_4, v_6)\}$, $\{(h_1, v_4), (h_2, v_5), (h_3, v_6), (h_4, v_7)\}$. Thus, this CRBM is defined by 4 pairwise weights. Each additional group would have its own set of weights and would contribute an additional 4 parameters. Thus the weight matrix for a CRBM is $N_H \times K$ where N_H is the length of the filters (number of visible units that each hidden unit is connected to) and K is the number of groups. In this example, $N_H = 4$ and $K = 1$.

inference (such as bottom-up and top-down inference). Max-pooling refers to operations where a local neighborhood of feature detection outputs is shrunk to a pooling node by computing the maximum of the local neighbors. Max-pooling makes the feature representation more invariant to local translations in the input data, and has been shown to be useful in computer vision [54, 40, 8, 27]. The energy function of the probabilistic max-pooling CRBM is defined as follows:

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}) &= -\sum_{k=1}^{K} \sum_{i,j=1}^{N_H} h_{ij}^k (\tilde{W}^k * \mathbf{v})_{ij} + \sum_{r,s=1}^{N_V} \frac{1}{2} v_{rs}^2 - \sum_{k=1}^{K} b^k \sum_{i,j=1}^{N_H} h_{ij}^k - c \sum_{r,s=1}^{N_V} v_{rs} \\ \text{s.t.} \quad & \sum_{(i,j) \in B_\alpha} h_{ij}^k \leq 1, \quad \forall k, \alpha \end{aligned}$$

Here, $\tilde{W}^k$ refers to flipping the original filter W^k in both upside-down and left-right directions, and $*$ denotes convolution. B_α refers to a block of locally neighboring hidden units (*i.e.*, pooling region) $h_{i,j}^k$ that are pooled to a pooling node p_α^k. The CRBM can be trained by approximately maximizing the log-likelihood of the unlabeled data via contrastive divergence [30], which has been successfully applied in optimizing many undirected graphical models that have intractable partition functions [82, 97, 31].

Since the model is highly over-complete, it is necessary to regularize [73, 79] it to prevent it from learning trivial or uninteresting feature representations. Specifically, it is typical to train the CRBM with a sparsity penalty term added to the log-likelihood objective to encourage each hidden unit group to have a mean activation close to a small constant. This can be implemented this with the following simple update rule (following each contrastive divergence update):

$$\Delta b_k \propto p - \frac{1}{N_H^2} \sum_{i,j} P(h_{ij}^k = 1 | \mathbf{v}), \tag{3.1}$$

where p is a target sparsity. The learning rate for the sparsity updates is chosen to make the hidden group's average activation (over the entire training set) close to the target sparsity, while allowing variations of activations depending on specific inputs. Table 3.1 overviews the steps for learning a CRBM; for details on learning and inference see [54, 53].

One of the attributes of CRBMs (similar to RBMs) is the ability to stack them where the pooling layer activation of one CRBM can be used as input to further train the next layer CRBM. In such a layered network (called a convolutional deep belief network, CDBN), each layer encodes statistical dependencies in the units in the layer below. CDBNs and related unsupervised learning algorithms such as auto-encoders [5] and sparse coding [73, 52] have been used to learn higher-level feature representations from unlabeled data, where different layers of the architecture capture information about the data at different scales.

3.2 Congealing with CRBMs

In the *deep congealing* algorithm [37, 33], a feature representation is first learned by training a CRBM (or CDBN) using the original data. Then the congealing algorithm is used to iteratively transform each instance to reduce the total entropy over the outputs of the CRBM (or CDBN) applied to all the data. We discuss the two steps of feature learning and alignment in more detail in the following subsections.

3.2.1 Feature Learning

Congealings optimization is based on conditional maximization which takes small steps in (transformation) parameter space to reduce its entropy (or variance) objective function. Consequently, a smooth optimization landscape is needed to escape local optima and converge to a reasonable alignment. Unfortunately, there is no

Algorithm: Learning convolutional RBM

Input: M curves of length N_V, $\mathbf{v}_{M \times N_V}$
Input: # of filters (or groups), K
Input: filter size, N_W
Input: pooling factor, C
Input: # of learning iterations, J
Input: # of samples per iteration, S
Output: K filter weights, $W_{K \times N_W}$
Output: K hidden unit biases, $b_{K \times 1}$
Output: visible unit bias, $c_{1 \times 1}$

$[W, b, c] = \text{initParams}(\mathbf{v}, K, N_W, C)$ — initialize parameters
for $j = 1$ to J { — main learning loop
  for $s = 1$ to S { — visible layer samples
    $v = \text{sample}(\mathbf{v})$ — sample visible layer
    $[W, b, c] = \text{contrastiveDivergenceUpdate}(v, W, b, c)$
  }
  if $j\%20 == 0$ {
    $[W, b] = \text{reorderFilters}(\mathbf{v}, W, b, c, N_W, S)$ — topological reordering
  }
}
return W, b, c

Table 3.1: Top-level pseudo-code for learning a convolutional RBM.

topology on the filters produced using a standard learning of a CRBM which may be problematic for congealing.

Therefore, we would like to learn filters with a linear topological ordering, such that when a particular pooling unit p_α^k at location α and associated with filter k is activated, the pooling units at the same location, associated with nearby filters, *i.e.*, $p_\alpha^{k'}$ for k' close to k, will also have partial activation. To learn a topology on the learned filters, we add the following group sparsity penalty to the learning objective function:

$$\mathcal{L}_{\text{sparsity}} = \lambda \sum_{k,\alpha} \sqrt{\sum_{k'} \omega_{k'-k} (p_\alpha^k)^2}$$

where ω_d is a Gaussian weighting, $\omega_d \propto \exp(-\frac{d^2}{2\sigma^2})$.

Let the term *array* be used to refer to the set of pooling units associated with a particular filter, *i.e.*, p_α^k for all locations α. This regularization penalty is a sum (L^1 norm) of L^2 norms, each of which is a Gaussian weighting, centered at a particular array, of the pooling units across each array at a specific location. In practice, rather than weighting every array in each summand, we use a fixed kernel covering five consecutive filters, *i.e.*, $\omega_d = 0$ for $|d| > 2$.

The rationale behind such a regularization term is that, unlike an L^2 norm, an L^1 norm encourages sparsity. This sum of L^2 norms thus encourages sparsity at the group level, where a group is a set of Gaussian weighted activations centered at a particular array. Therefore, if two filters are similar and tend to both activate for the same visible data, a smaller penalty will be incurred if these filters are nearby in the topological ordering, as this will lead to a more sparse representation at the group L^2 level. To account for this penalty term, we augment the learning algorithm by taking a step in the negative derivative with respect to the CRBM weights. We define $\alpha(i, j)$ as the pooling location associated with position (i, j), and J as

$$J_{ij}^{k,k'} = \frac{1}{\sqrt{\sum_{k''} \omega_{k''-k'} (p_{\alpha(i,j)}^{k''})^2}} p_{\alpha(i,j)}^k (1 - p_{\alpha(i,j)}^k) h_{ij}^k.$$

We can write the full gradient as

$$\nabla_{W^k} \mathcal{L}_{\text{sparsity}} = \lambda \sum_{k'} \omega_{k-k'} (v * \tilde{J}^{k,k'}),$$

where $*$ denotes convolution and $\tilde{J}^{k,k'}$ means $J^{k,k'}$ flipped horizontally and vertically. Thus we can efficiently compute the gradient as a sum of convolutions.

Following the procedure given by Sohn *et al.* [88], we initialize the filters using expectation-maximization under a mixture of Gaussians / Bernoulli, before proceeding with CRBM learning. Therefore, when learning with the group sparsity penalty,

we periodically reorder the filters using the following greedy strategy: taking the first filter, we iteratively add filters one by one to the end of the filter set, picking the filter that minimizes the group sparsity penalty.

3.2.2 Alignment Objective Function

After training a CRBM, we can compute the posterior of the pooling units given the input data. Specifically, letting $I(h_{ij}^k) \triangleq b^k + (\tilde{W}^k * \mathbf{v})_{ij}$, we can infer the pooling unit activations as a softmax function:

$$P(p_\alpha^k = 1|\mathbf{v}) = \frac{\sum_{(i',j')\in B_\alpha} \exp(I(h_{i'j'}^k))}{1 + \sum_{(i',j')\in B_\alpha} \exp(I(h_{i'j'}^k))}$$

Given a set of poorly aligned curves, our goal is to iteratively transform each curve to reduce the total entropy over the pooling layer outputs of a CRBM applied to each of the curves. For a CRBM with K pooling layer groups, we now have K location stacks at each curve location (after max-pooling), over a binary distribution for each location stack. Given N unaligned curves, let P be the number of pooling units in each group in the CRBM. We use the pooling unit probabilities, with the interpretation that the pooling unit can be considered as a mixture of sub-units that are on and off [49]. Letting $p_\alpha^{k,(n)}$ be the pooling unit α in group k for curve n under some transformation T^n, we define $D_\alpha^k(1) = \frac{1}{N}\sum_{n=1}^N p_\alpha^{k,(n)}$ and $D_\alpha^k(0) = 1 - D_\alpha^k(1)$. Then, the entropy for a specific pooling unit is

$$H(D_\alpha^k) = -\sum_{s\in\{0,1\}} D_\alpha^k(s)\log(D_\alpha^k(s)). \tag{3.2}$$

Consequently a good measure for alignment would be summing the entropy of the pooling units across all the groups: $\sum_{k=1}^K \sum_{\alpha=1}^P H(D_\alpha^k)$. Such a similarity would be sufficient for many types of data (and this was the one used for the image alignment experiments in § 3.4). However, for curve congealing we found this to not be sufficient

since for some curves, none of the filters activated in large (uneventful) locations. This led to some of the curves being unnecessarily transformed in those locations since it did not change the entropy of the pooling units. To counteract this, we used two objective functions in our modified curve congealing algorithm: the sum of the entropy of pooling unit activations (introduced above) and the sum of location-wise variance of the raw curves (used in Chapter 2). The latter objective ensured that no transformation was inflicted on the data that unnecessarily increased the variance of the data set. So in our modified congealing algorithm, a transformation parameter update is accepted if it either decreases the entropy of the pooling activations or the variance of the raw curves, but without increasing the other. We found that using both objectives performed better than using either in isolation.

3.3 Experiment: Curve Alignment using CRBMs

We experimented with curve alignment using CRBM features and evaluated its effect on curve classification as we did in the previous chapter. Preferably, we would have trained a separate CRBM for each of the 13 data sets in the UCR repository we evaluated on. However, given the small number of training examples in many data sets (as few as 24), we opted to train a single CRBM using all the *training* examples of all the data sets. Since the number of training examples for each data set ranges from 24 to 300, we replicated the examples from each data set to ensure that each data set contributed at least 200 curves. For example, if a data set had 30 training examples, we sampled 170 times (with replacement) and for each curve sample we randomly transformed it to produce a slight variation. This provided us with a data set of 2700 curves that was used to train the CRBM. Furthermore, to speed-up and simplify the learning, we rescaled all the curves to a length of 100 since different data sets have different length curves. Following the parameters in our earlier work of

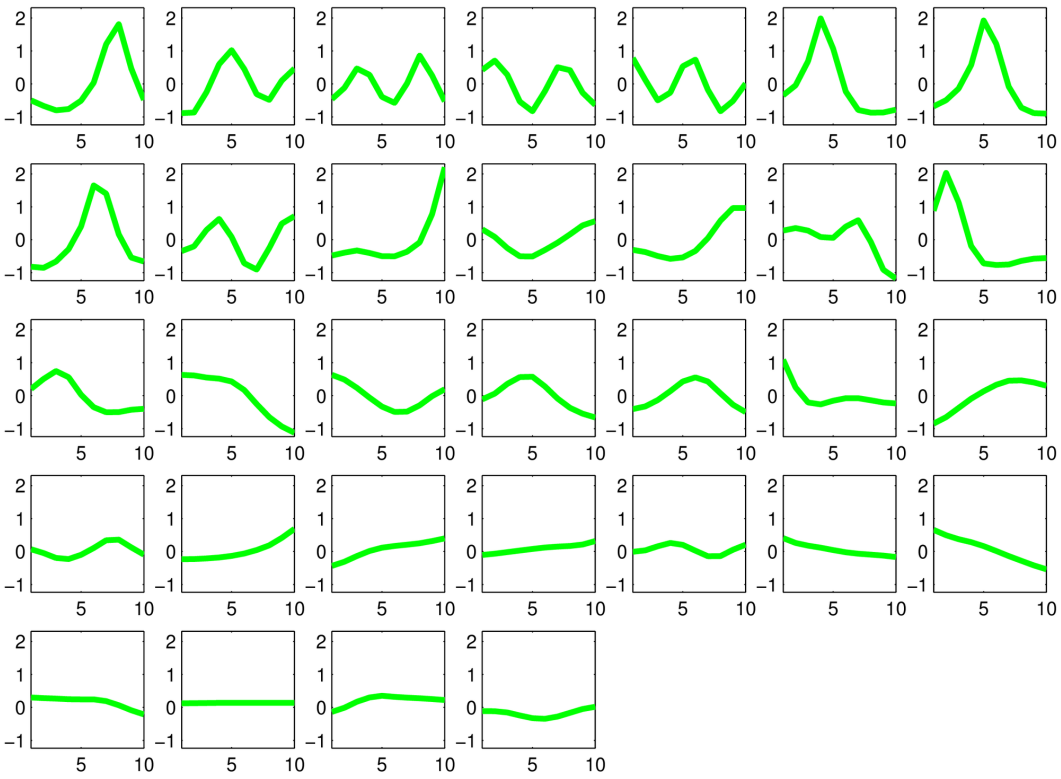

Figure 3.2: The 32 CRBM filters learned using 2700 curves from the UCR time series repository. The filter size is 10 and the pooling area is 5. The filters are ordered (row-major order) based on their mean pooling unit activations across all the curves (high mean activations first).

applying CRBMs to face images (summarized in § 3.4), we learned 32 filters of length 10 and a pooling block of 5.

Figures 3.2, 3.5, 3.6, 3.7, 3.8, and 3.9 provide visualizations for better understanding the filters learned and their activations on different curves and data sets. Figure 3.2 displays the filters (i.e. visualizing the weight matrix W^k of each group). Figure 3.5 shows the pooling unit activations for different filters on different curves and data sets. The figure highlights that the majority of filters activate on most of the curves and data sets and no data set-specific filters were learned. Figures 3.6, 3.7, 3.8, and 3.9 show the hidden unit activations with highest probability for

1	2	3	4	5	6
Alignment	None		Raw	CRBM	
Classification	Raw	CRBM	Raw		CRBM
Coffee	96.43	**100.00**	**100.00**	**100.00**	**100.00**
Beef	83.33	**86.67**	73.33	80.00	80.00
OliveOil	**86.67**	83.33	**86.67**	**86.67**	**86.67**
FaceFour	87.50	89.77	**94.32**	93.18	93.18
Lighting2	68.85	57.38	67.21	**73.77**	**73.77**
Lighting7	67.12	56.16	64.38	**75.34**	**75.34**
ECG200	81.00	80.00	83.00	84.00	**85.00**
Trace	77.00	**99.00**	97.00	86.00	94.00
Gun_Point	90.00	93.33	99.33	**100.00**	**100.00**
FISH	84.00	92.00	94.29	95.43	**97.14**
OSULeaf	46.28	**59.50**	49.59	54.96	57.85
synthetic_control	92.00	**99.00**	95.33	96.00	98.00
CBF	88.44	86.89	97.56	**97.78**	**97.78**
Mean	80.66	83.31	84.77	86.39	**87.59**

Table 3.2: Curve classification using a linear SVM. The first column contains the name of the UCR data set (one per row) and the remaining five columns contain the classification performance using different settings which are indicated by the rows labeled "Alignment" and "Classification" which specify the feature representation used for each task. For example, column 5 contains the classification accuracies when alignment was performed using the CRBM features and classification was performed using the raw curves. For all the experiments that include alignment (columns 4 - 6), the input to the SVM classifier was the concatenation of the original curve, the transformed curve, and the transformation parameters (i.e. the simple concatenation method used in Chapter 2).

each filter. One interesting observation is that the curves (for each filter) are quite similar, highlighting the repeatability of the filter activations.

We performed a series of alignment and classification experiments to evaluate the utility of the CRBM representation for both tasks. Table 3.2 includes the classification results for different settings and Figure 3.3 shows a scatter plot highlighting the improvements to a linear SVM classifier that can be attained by aligning and classifying the curve data sets using the CRBM feature representation. When classifying with joint alignment, we concatenate the features from the original curve, aligned

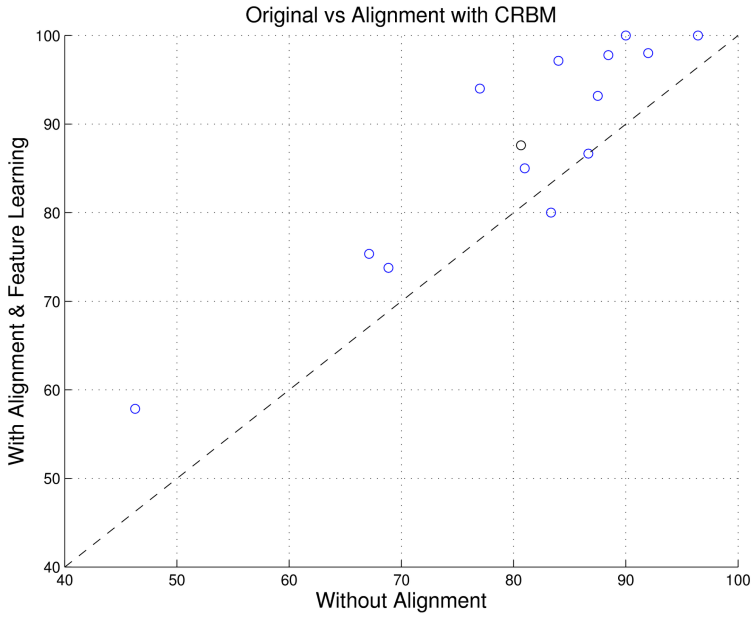

Figure 3.3: Classification scatter plots comparing a linear SVM without alignment or feature learning (column 2 in Table 3.2) to using both alignment and classification with CRBM representation (column 6 in Table 3.2).

curve and transformation parameters as we did in the previous chapter. The only difference is that for the results in column 6, the CRBM representation was not used for both the original and aligned curves because the dimensionality of the resulting vector would be too large (576 pooling activations for each of the original and aligned curves as well as 20 transformation parameters results in a 1172 dimensional vector) for the data sets that have few training examples. Consequently for data sets that had fewer than 100 training examples we used the raw curve representation when classifying and for data sets with more than 100 training examples we used the CRBM representation (note that for all data sets, the CRBM representation was used for alignment). Overall, we found that joint alignment and unsupervised feature learning can improve the performance of a linear SVM classifier by approximately **7**% averaged over the 13 data sets.

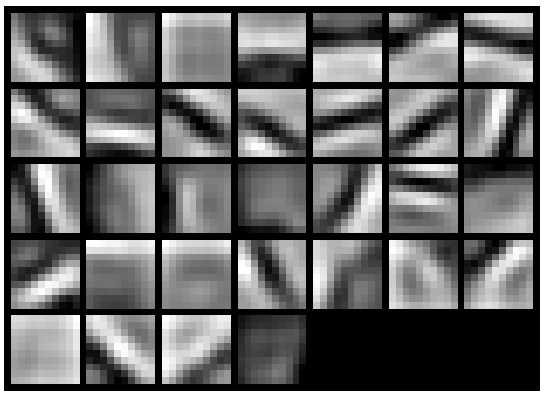

Figure 3.4: Filters learned from a CRBM on LFW images.

3.4 Experiment: Face Alignment with CRBMs

One of our motivations for using CRBMs as our framework for feature learning is its applicability to many types of data. To showcase this versatility, we highlight earlier results on improving the alignment of face images [37]. Specifically, the task we consider here is face verification where bringing the images into correspondence can improve classification performance. To obtain verification accuracy, we use a variation on the method of Cosine Similarity Metric Learning (CSML) [72], one of the top-performing methods on LFW. We evaluate the performance of the CSML classifier on view 1 of the LFW data set [36, 38] using images aligned with three different methods: 1. No alignment, 2. Congealing with SIFT features [34], and 3. Congealing with CRBM features [37]. For the CRBM set-up, 32 10×10 filters were learned with a pooling block of 5×5 (see Figure 3.4).

The baseline verification accuracy without alignment was 74.2%. With SIFT congealing the accuracy improved to 75.8 and with CRBM congealing the accuracy improved to 82.0%. These results not only highlight the utility of unsupervised joint alignment for improving face verification, but also the effectiveness of features learned using a CRBM (particularly for the task of face alignment).

To offer some insight into the type of information captured by a CRBM trained on face images, we clustered the training set (3443 images) with KMeans (20 clusters)

using the CRBM output as the representation for each image. Figures 3.10 and 3.11 show the 8 images closest to the centroid of each of the 20 clusters. The clusters mostly split on pose (left-facing, right-facing, and center) and gender/hair, while cluster 16 was reserved for those wearing hats.

3.5 Summary

We presented a modification to the congealing framework that incorporates features learned from a convolutional restricted Boltzmann machine. We showed that using the *exact* same unsupervised feature learning framework can improve the alignment of both curve and face images.

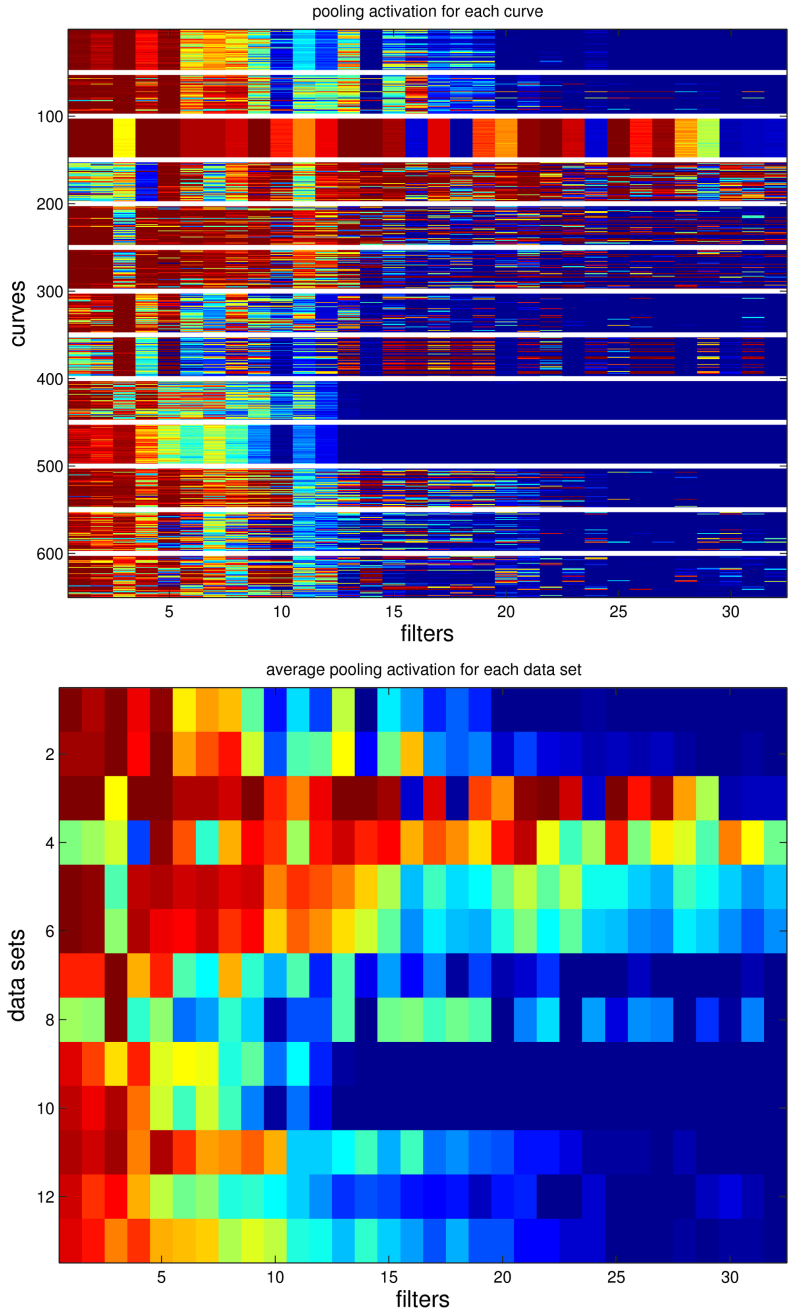

Figure 3.5: CRBM pooling unit activations (the colors follow a standard scaled-image range where dark blue represents low activations and dark red represents high activations). For both plots, the filters are ordered from left to right by the magnitude of their activations (same ordering as in Figure 3.2). **Top:** Displays a matrix of the maximum pooling activation (for each of the 32 filters) on 50 randomly selected curves from each of the 13 data sets. The curves from each data set are grouped together and separated by a white line. **Bottom:** Similar to top, but we average the maximum pooling activation for each filter across all the curves in a data set. This provides us with a measure for how often a specific filter activates for each data set.

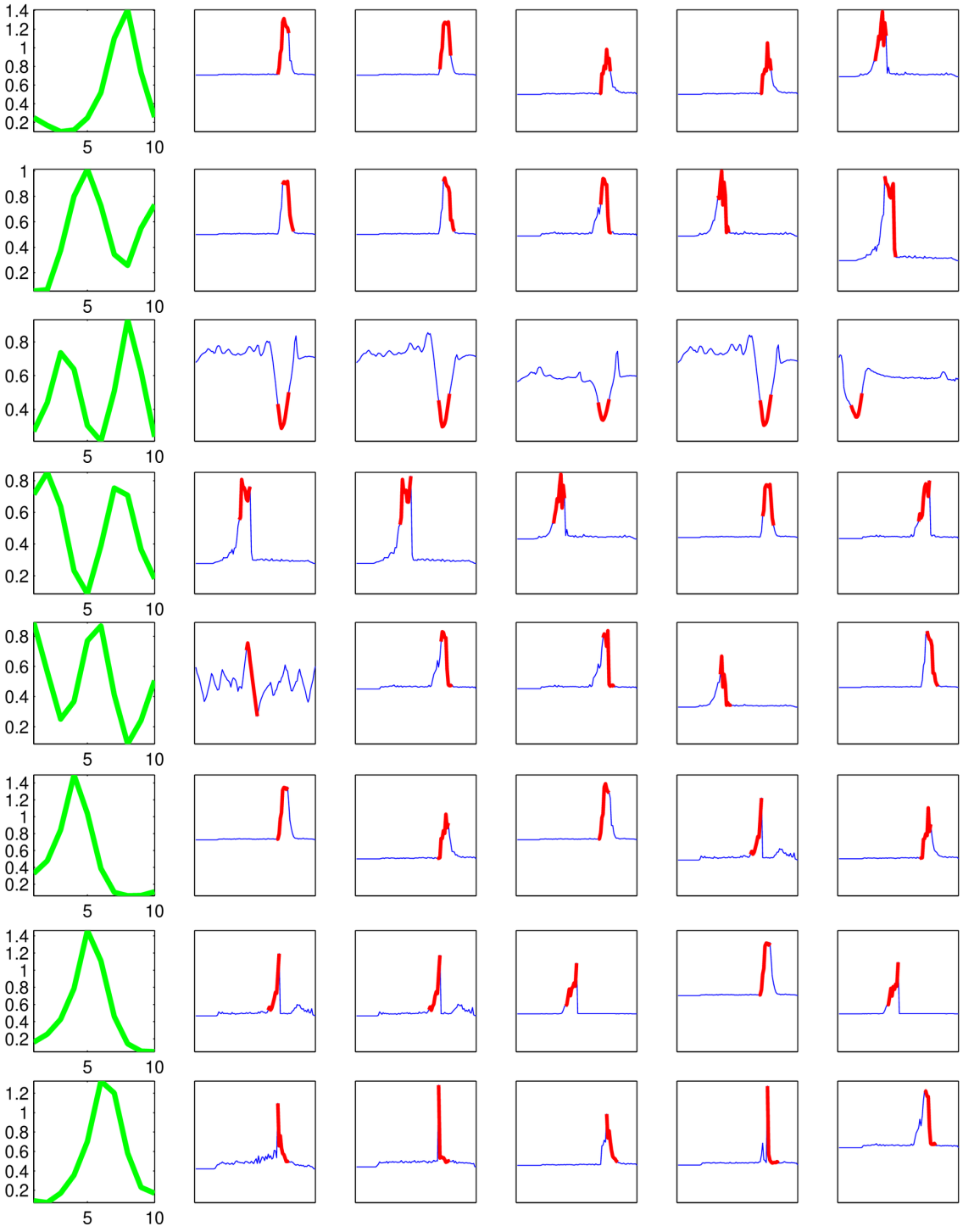

Figure 3.6: CRBM filters (1 - 8) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation).

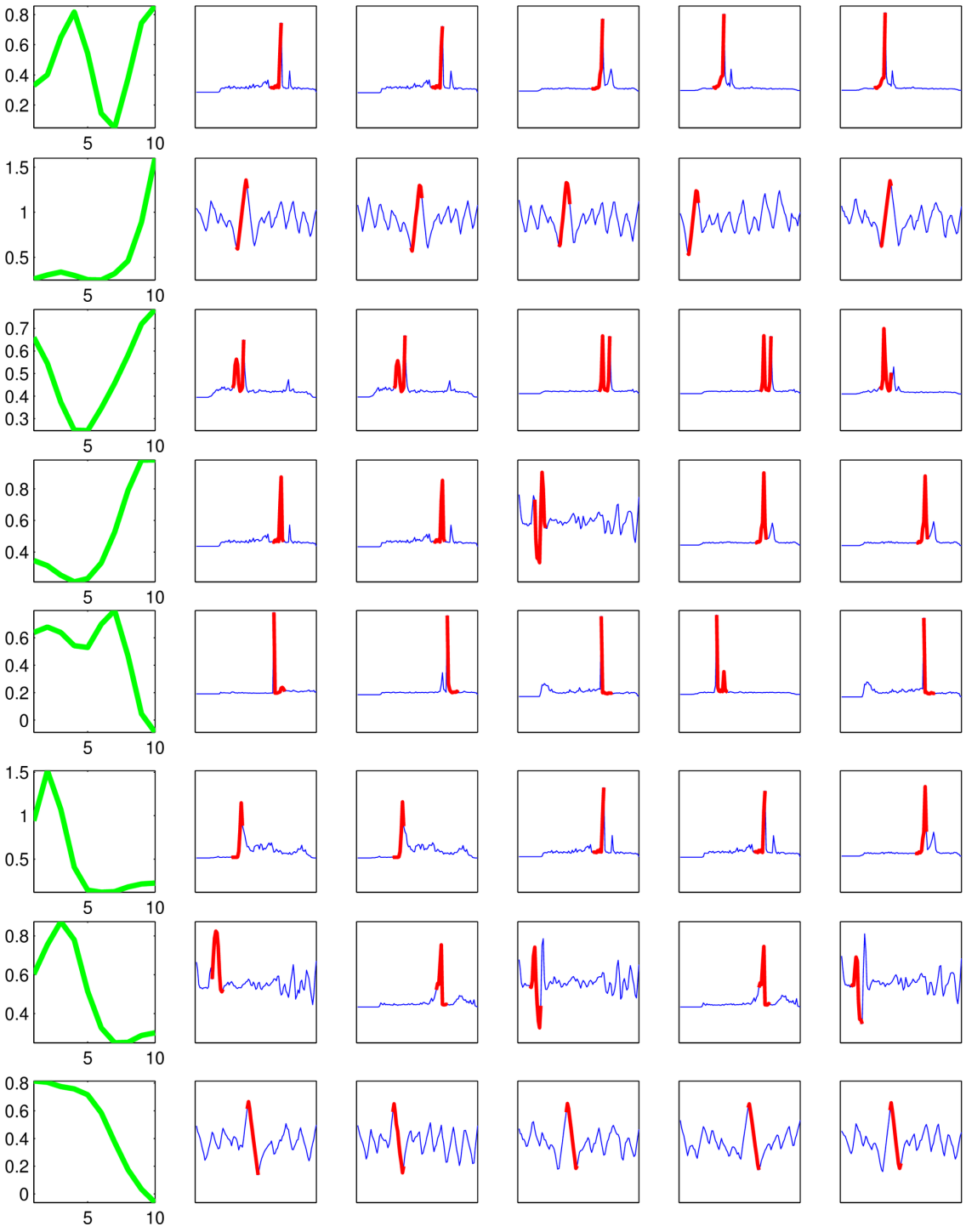

Figure 3.7: CRBM filters (9 - 16) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation).

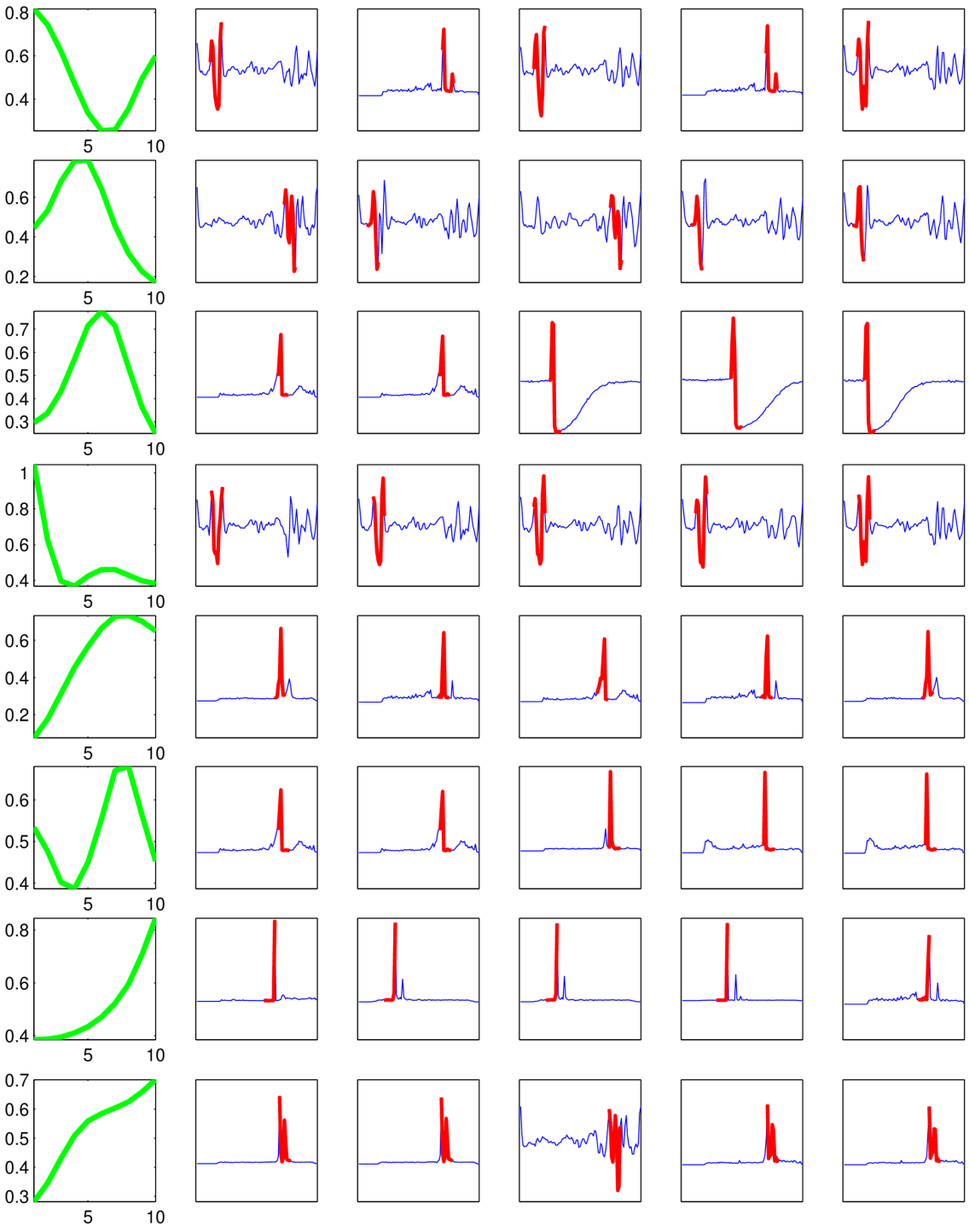

Figure 3.8: CRBM filters (17 - 24) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation).

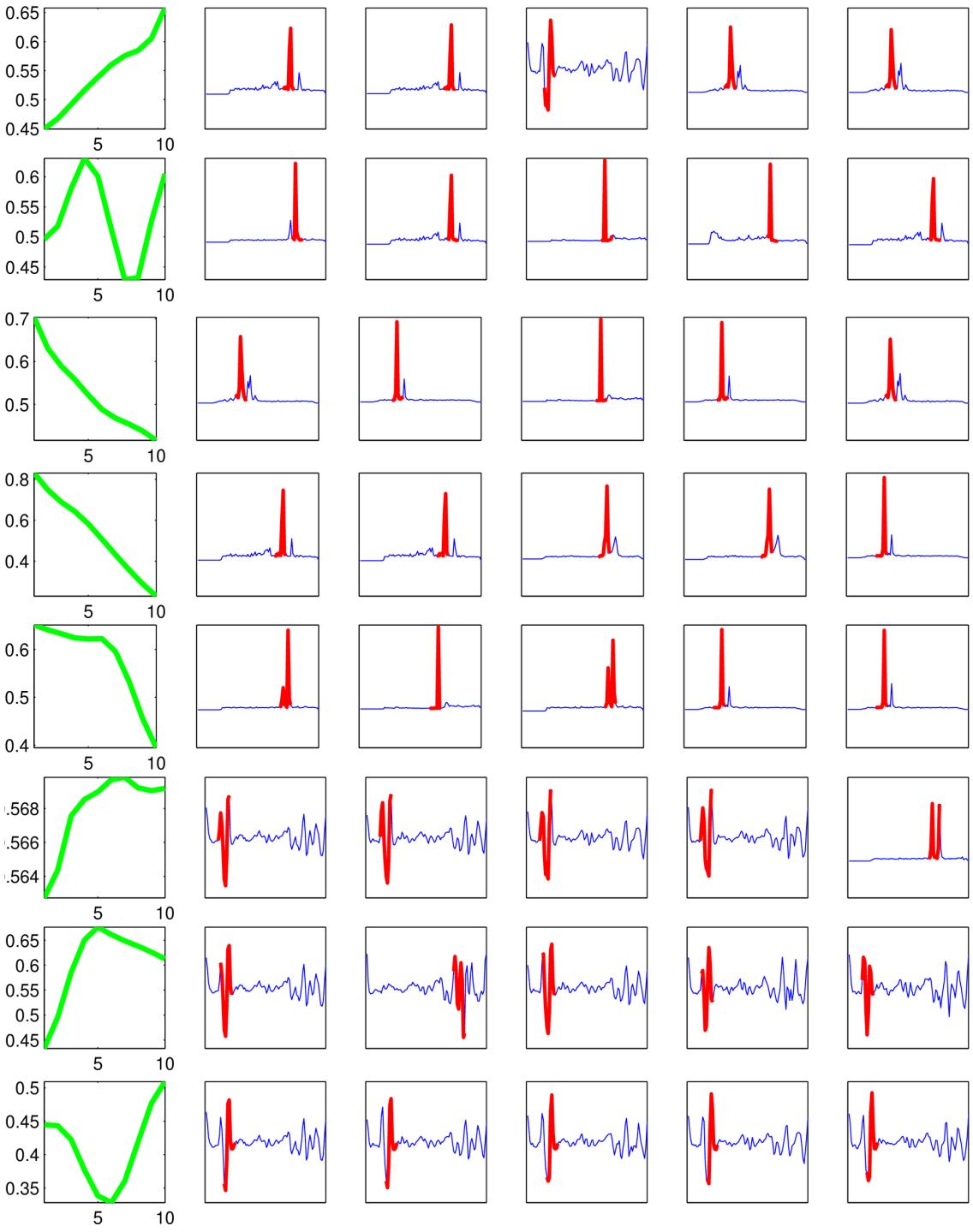

Figure 3.9: CRBM filters (25 - 32) and their top hidden unit activations. Each row displays one of the 32 filters and the 5 curves with highest hidden unit activation (the overlaid red segment in each curve represents the location of the highest activation).

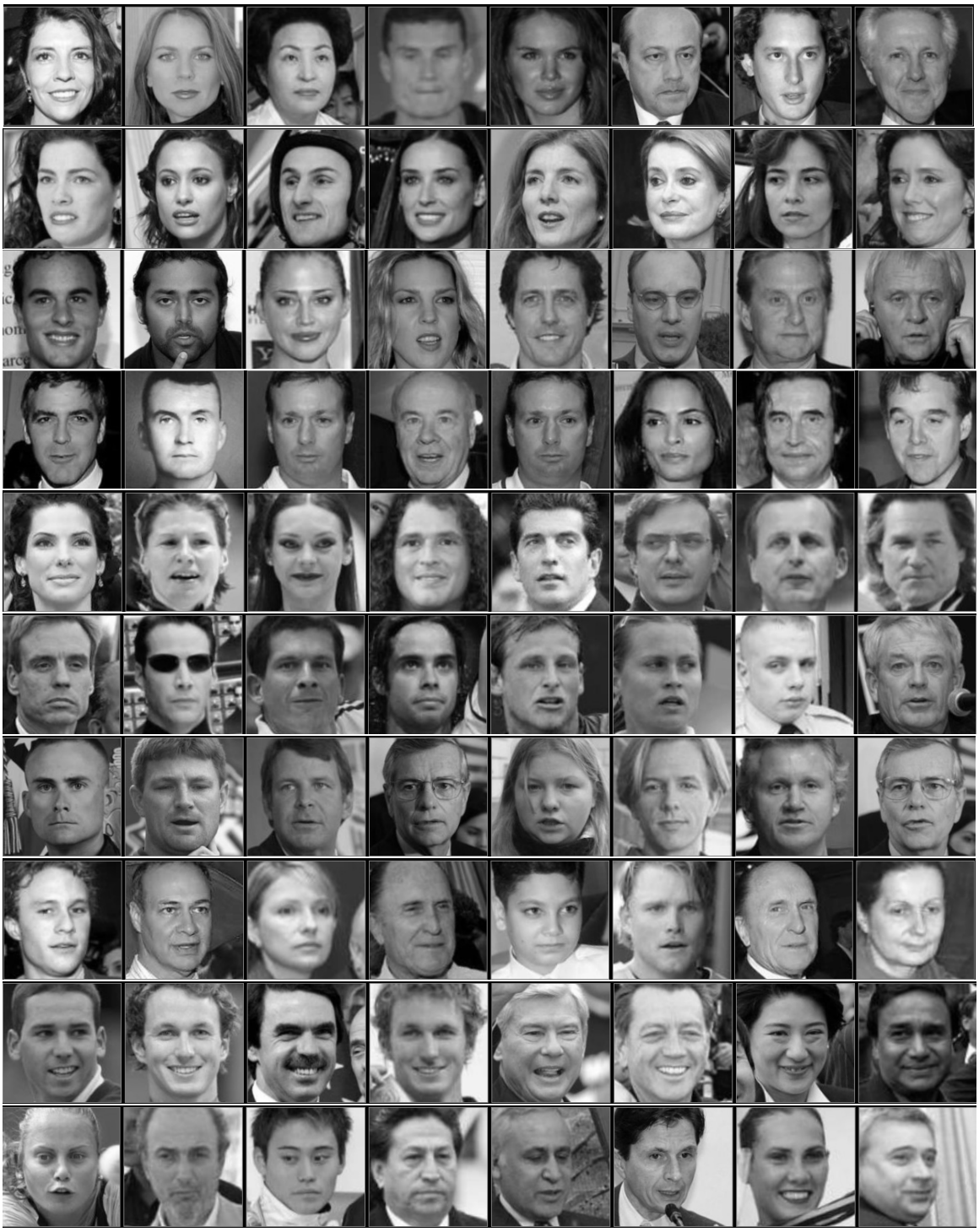

Figure 3.10: Clusters 1-10 of 20 resulting from running KMeans on the 3443 training images using the CRBM representation. Each row contains the eight images from each cluster that are closest (in Euclidean distance) to the cluster centroid.

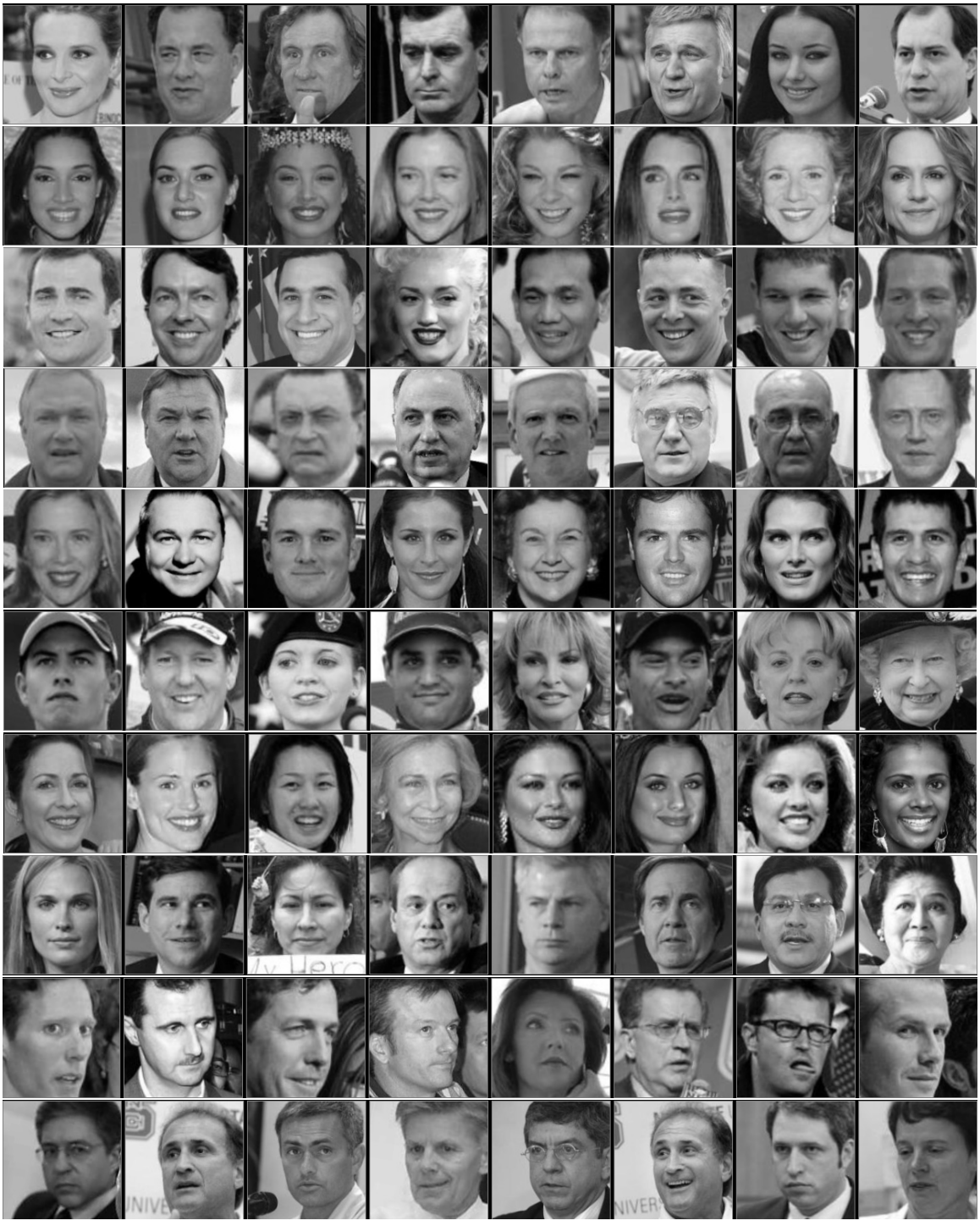

Figure 3.11: Clusters 11-20 of 20 resulting from running KMeans on the 3443 training images using the CRBM representation. Each row contains the eight images from each cluster that are closest (in Euclidean distance) to the cluster centroid.

CHAPTER 4

JOINT ALIGNMENT AND CLUSTERING

The problem addressed in this chapter is joint alignment of a data set that may contain multiple groups or clusters. Previous nonparametric alignment algorithms (e.g. congealing [49]) typically fail to acknowledge the multi-modality of the data set resulting in poor performance on complex data sets. We address this by simultaneously aligning and clustering [21, 22, 59] the data set. As we will show (and illustrated in Figure 4.2), solving both alignment and clustering simultaneously offers many advantages over first clustering the data set and then aligning the instances in each cluster [1].

To this end, we developed a nonparametric [2] Bayesian joint alignment and clustering model that is a generalization of the standard Bayesian infinite mixture model. Our model possesses many of the favorable characteristics of congealing, while overcoming its drawbacks. More specifically, it:

- Explicitly clusters the data which provides a mechanism for handling complex data sets. Furthermore, the use of a Dirichlet process prior enables learning the number of clusters in a data-driven fashion.

- Can use any generic transformation function parameterized by a vector. This decouples our model from the specific transformations which allows us to plug in different transformation operations for different data types.

---

[1] This work was published in the 2012 Conference on Uncertainty in Artificial Intelligence [63].

[2] Here, we use the term nonparametric to indicate that the number of model parameters can grow (a property of the infinite mixtures), and not that the distributions are not parametric.

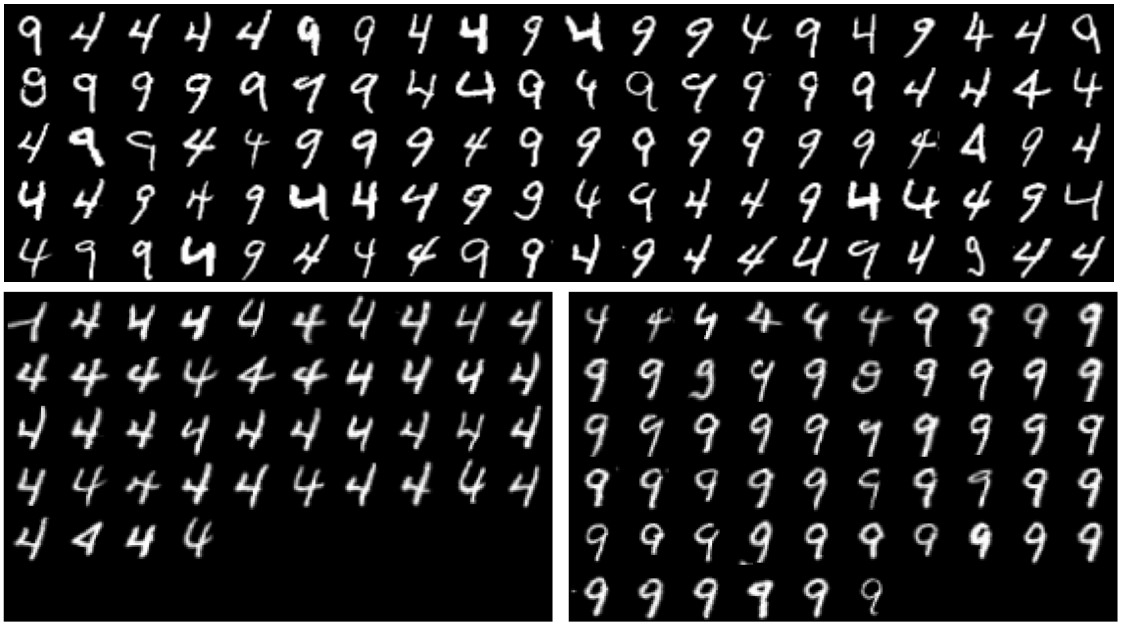

Figure 4.1: Joint alignment and clustering: given 100 unlabeled images (top), without any other information, our algorithm (§ 4.2) chooses to represent the data with two clusters, *aligns* the images and *clusters* them as shown (bottom). Our clustering accuracy is 94%, compared to 54% with K-means using two clusters (using the minimum error across 200 random restarts). Our model is not limited to affine transformations or images.

- Enables the encoding of prior beliefs regarding the degree of variability in the data set, as well as regularizes the transformation parameters in a principled way by treating them as random variables.

We first present a Bayesian joint alignment model (§ 4.1) that assumes a unimodal data set (i.e. only one cluster). This model is a special case of our proposed joint alignment and clustering model that we introduce in § 4.2. We then discuss several variations of our model in § 4.3 and conclude in § 4.4 with directions for future work.

Problem Definition. We are provided with a data set $\mathbf{x} = \{x_i\}_{i=1}^N$ of N items and a transformation function, $x_i = \tau(y_i, \rho_i)$ parameterized by ρ_i. Our objective is to recover the set of transformation parameters $\{\rho_i\}_{i=1}^N$, such that the aligned data set $\{y_i = \tau(x_i, \rho_i^{-1})\}_{i=1}^N$ is more coherent. In the process, we also learn a clustering

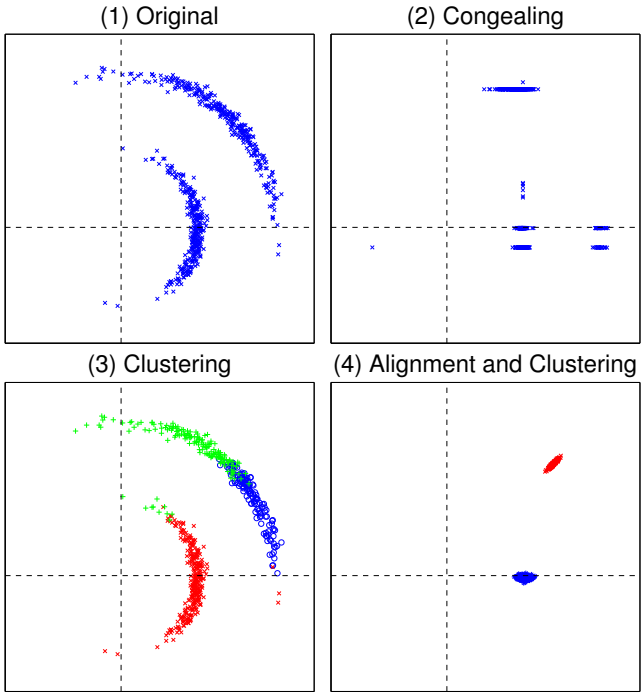

Figure 4.2: Illustrative example. (1) shows a data set of 2D points. The set of allowable transformations is rotations around the origin. (2) shows the result of the congealing algorithm which transforms points to minimize the sum of the marginal entropies. This independence assumption in the entropy computation causes the points to be squeezed into axis aligned groups. (3) highlights that clustering alone with an infinite mixture model may result in a larger number of clusters. (4) shows the result of the model presented in this chapter. It discovers two clusters and aligns the points in each cluster correctly.

assignment $\{z_i\}_{i=1}^N$ of the data points. Here ρ_i^{-1} is defined as the parameter vector generating the inverse of the transformation that would be generated by the parameter vector ρ (i.e. $x_i = \tau(\tau(x_i, \rho_i^{-1}), \rho_i)$).

4.1 Bayesian Joint Alignment

The Bayesian alignment (BA) model assumes a unimodal data set (in § 4.2 this assumption is relaxed). Consequently there is a single set of parameters (θ and ρ) that generate the entire data set (see Figure 4.3). Under this model, every observed data item, x_i, is generated by transforming a canonical data item, y_i, with transformation,

ρ_i. More formally, $x_i = \tau(y_i, \rho_i)$, where $y_i \sim F_D(\theta)$ and $\rho_i \sim F_T(\varphi)$. The auxiliary variable y_i is not shown in the graphical model for simplicity. Given the Bayesian setting, the parameters θ and φ are random variables, with their respective prior distributions, $H_D(\lambda)$ and $H_T(\alpha)$.[3]

The model does not assume that there exists a single perfect canonical example that explains all the data, but uses a parametric distribution $F_D(\theta)$ to generate a slightly different canonical example, y_i, for each data item, x_i. This enables it to explain variability in the data set that may not be captured with the transformation function alone. The model treats the transformation function as a black-box operation, making it applicable to a wide range of data types (e.g. curves, images, and 3D MRI scans), as long as an appropriate transformation function is specified.

For both this model and the full joint alignment and clustering model introduced in the next section we use exponential family distributions for $F_D(\theta)$ and $F_T(\varphi)$ and their respective conjugate priors for $H_T(\alpha)$ and $H_D(\lambda)$. This allows us to use Rao-Blackwellized sampling schemes [9] by analytically integrating out the model parameters and caching sufficient statistics for efficient likelihood computations. Furthermore, the hyperparameters now play intuitive roles where they act as a pseudo data set and are easier to set or learn from data.

4.1.1 Learning

The model likelihood factorizes based on the graph structure (Figure 4.3) in the following way:

[3] Here we assume that the hyperparameters α and λ are fixed, but they can be learned or sampled if necessary.

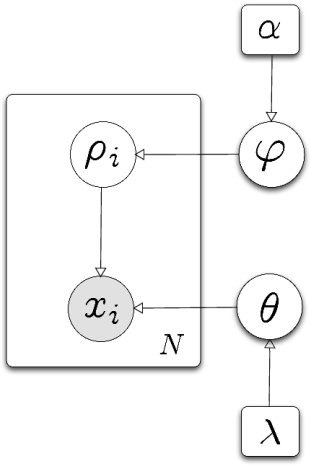

Figure 4.3: Graphical representation for our proposed Bayesian alignment model (§ 4.1)

$$
\begin{aligned}
p(\mathbf{x}, \boldsymbol{\rho}, \varphi, \theta\,;\, \alpha, \lambda) &= p(\varphi;\alpha)p(\theta;\lambda)p(\boldsymbol{\rho}|\varphi)p(\mathbf{x}|\boldsymbol{\rho},\theta) \\
&= p(\varphi;\alpha)p(\theta;\lambda)p(\boldsymbol{\rho}|\varphi)p(\tau(\mathbf{x},\boldsymbol{\rho}^{-1})|\theta) \\
&= p(\varphi;\alpha)p(\theta;\lambda)\prod_{i=1}^{N} p(\rho_i|\varphi)p(\tau(x_i,\rho_i^{-1})|\theta) \\
&= p(\varphi;\alpha)p(\theta;\lambda)\prod_{i=1}^{N} p(\rho_i|\varphi)p(y_i|\theta)
\end{aligned}
$$

where $y_i = \tau(x_i, \rho_i^{-1})$.

Given a data set $\{x_i\}_{i=1}^N$ we wish to learn the parameters of this model (θ, φ) and the hidden variables ($\{\rho_i\}_{i=1}^N$). A standard Gibbs sampler would iterate over all the variables and sample from the conditional for each.

However, when conjugate priors are used it is possible to integrate out the model parameters (θ and φ) and only sample the hidden variables ($\{\rho_i\}_{i=1}^N$). Such Rao-Blackwellized sampling schemes [9] typically speed up convergence. The intuition is that the model parameters are implicitly updated with the sampling of every transformation parameter instead of once per Gibbs iteration. The resulting Gibbs sampler only iterates over the transformation parameters:

$$
\begin{aligned}
\forall_{i=1:N}\ \rho_i^{(t)} &\sim p(\rho_i|\mathbf{x}, \boldsymbol{\rho}_{-i}^{(t)}; \alpha, \lambda) \\
&\propto p(\rho_i, x_i|\mathbf{x}_{-i}, \boldsymbol{\rho}_{-i}^{(t)}; \alpha, \lambda) \\
&= p(x_i|\rho_i, \mathbf{x}_{-i}, \boldsymbol{\rho}_{-i}^{(t)}; \lambda) p(\rho_i|\boldsymbol{\rho}_{-i}^{(t)}; \alpha) \\
&= p(y_i|\mathbf{y}_{-i}^{(t)}; \lambda) p(\rho_i|\boldsymbol{\rho}_{-i}^{(t)}; \alpha).
\end{aligned}
$$

The superscript (t) in the above equations refers to the Gibbs iteration number. $\boldsymbol{\rho}_{-i}$ refers to all the variables $\{\rho_i\}_{i=1}^N$ except ρ_i, and similarly for $\mathbf{y}_{-i}$.

Sampling ρ_i is complicated by the fact that $p(y_i|\mathbf{y}_{-i}^{(t)}, \lambda)$ depends on the transformation function. Previous alignment research [49, 59], has shown that the gradient of an alignment objective function with respect to the transformations provides a strong indicator for how alignment should proceed. One option would be Hamiltonian Monte Carlo sampling [70] which uses the gradient as a drift factor to influence sampling. However, instead of relying on direct sampling techniques, we use approximations based on the posterior mode [23]. Such an approach is more direct since it is expected that the distribution will be tightly concentrated around the mode. Thus, at each iteration the transformation parameter is updated as follows:

$$
\rho_i^{(t)} = \arg\max_{\rho_i} p(y_i|\mathbf{y}_{-i}^{(t)}; \lambda) p(\rho_i|\boldsymbol{\rho}_{-i}^{(t)}; \alpha).
$$

The maximization is performed using the Nelder-Mead method since it does not rely on partial derivates which may be difficult to compute for some transformation functions. Interestingly, the same learning scheme can be derived using the incremental variant [71] of hard-EM.

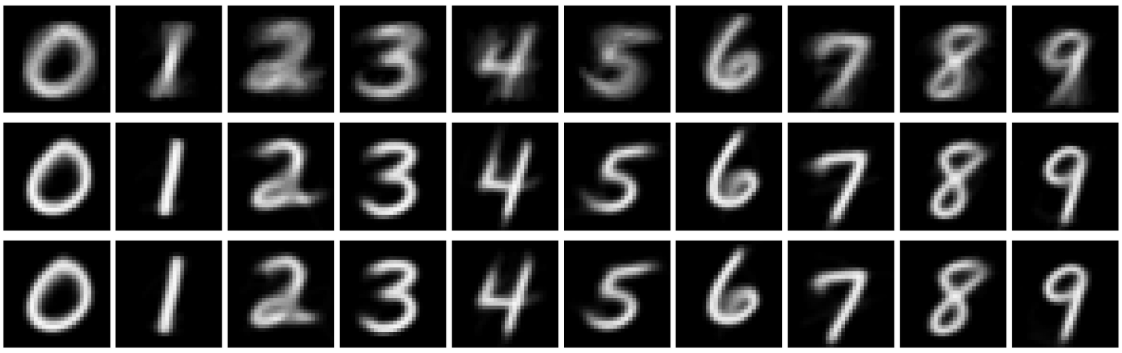

Figure 4.4: Comparison of congealing and Bayesian alignment on digits. Each image is the mean image of a specific digit class. **Top:** before alignment. **Middle:** alignment with congealing. **Bottom:** alignment with Bayesian alignment. Note that the qualitative differences between congealing and Bayesian alignment are negligible. Table 4.1 contains quantitative metrics.

4.1.2 Model Characteristics

The objective function optimized in our model contains two key terms, a data term, $p(\mathbf{x}|\boldsymbol{\rho}, \theta)$, and a transformation term, $p(\boldsymbol{\rho}|\varphi)$. The latter acts as a regularizer to penalize large transformations and prevent the data from being annihilated. One advantage of our model is that large transformations are penalized in a principled fashion. More specifically, the cost of a transformation, ρ_i is based on the *learned* parameter φ which depends on the transformations of all the other data items, $\boldsymbol{\rho}_{-i}$, and the hyperparameters, α. Learning φ from the data is a more effective means for assigning costs than handpicking them.

The model has several other favorable qualities. It is efficient, can operate on large data sets while maintaining a low memory footprint, allows continuous transformations, regularizes transformations in a principled way, is applicable to a large variety of data types, and its hyperparameters are intuitive to set. Its main drawback is the assumption of a unimodal data set, which we remedy in § 4.2. We first evaluate this model on digit and curve alignment.

Digit	Original		Congealing		Bayes. Align.	
	entropy	pdist	entropy	pdist	entropy	pdist
0	0.359	9.652	0.248	5.418	0.253	5.363
1	0.187	6.687	0.111	2.967	0.114	3.058
2	0.380	10.184	0.289	6.952	0.298	7.069
3	0.330	9.168	0.229	5.296	0.228	5.148
4	0.292	8.466	0.216	5.422	0.217	5.212
5	0.325	9.251	0.225	5.538	0.224	5.450
6	0.278	8.276	0.211	5.265	0.211	5.052
7	0.277	8.322	0.188	4.719	0.195	4.702
8	0.317	8.876	0.252	5.993	0.248	5.763
9	0.259	7.937	0.184	4.525	0.184	4.275
Mean	0.300	8.682	0.215	5.210	0.217	5.109

Table 4.1: Single-class digit alignment. We evaluated three techniques: Original (no alignment), Congealing (binary congealing [49]), and our proposed Bayesian alignment model. We computed two alignment scores: the mean entropy across each pixel stack, and the mean pairwise distance across all the images.

4.1.3 Experiments

We evaluate the Bayesian alignment model on both curve and image data sets, comparing it to congealing.

Digits. We selected 50 images of every digit class from the MNIST data set and performed alignment on each digit class independently. The mean images before and after alignment are presented in Figure 4.4. We allowed 7 affine image transformations: scaling, shearing, rotating and translation. $F_D(\theta)$ is the product of independent Bernoulli distributions, one for each pixel location, and $F_T(\varphi)$ is a $7-$D zero mean diagonal Gaussian. Table 4.1 show two alignment scores for each digit class for both congealing and Bayesian alignment. Overall, we conclude that the performance of Bayesian alignment matches congealing for binary images.

Curves. We ran our Bayesian alignment model on the 75 synthetic data sets generated in Chapter 2 to evaluate congealing. We used the same transformation function in curve congealing [64], which allows non-linear time warping (4 parameters), non-

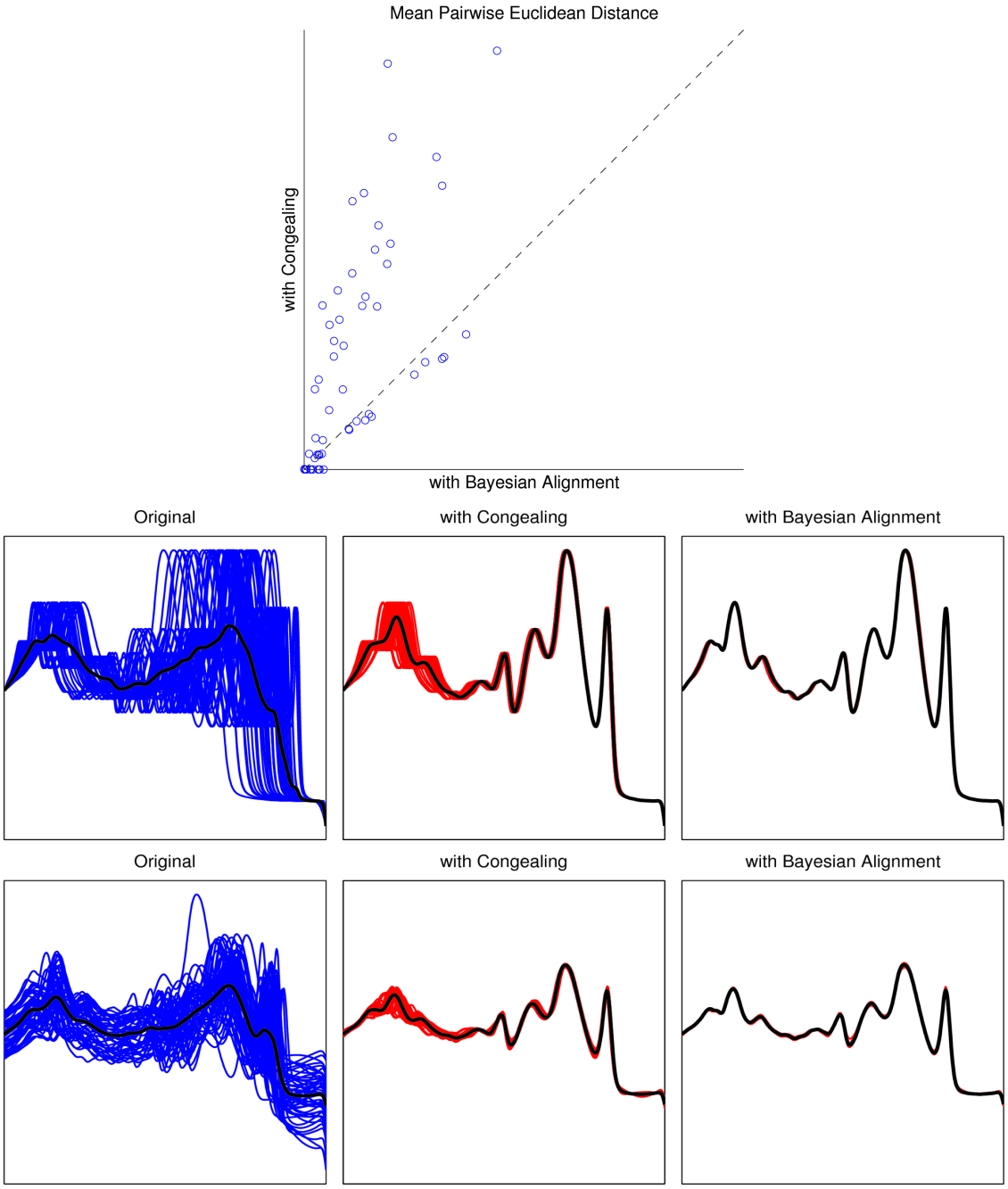

Figure 4.5: Comparison of congealing and Bayesian alignment on curve data sets. **Top:** scatter plot of the mean pairwise distance score of congealing and the Bayesian alignment algorithm across the 75 synthetic curve data sets (generated in Chapter 2). **Middle and Bottom:** Two of the examples from Chapter two were congealing got stuck in local minima.

linear amplitude scaling (14 parameters), linear amplitude scaling (1 parameter), and amplitude translation (1 parameter). $F_D(\theta)$ was set to a diagonal Gaussian distribution (i.e. we treat the raw curves as a random vector), and $F_T(\varphi)$ was a 20−D zero mean diagonal Gaussian. Figure 4.5 shows a scatter plot of mean pairwise distances for both congealing and Bayesian alignment, as well as sample alignment results on two challenging data sets, where Bayesian alignment is able to converge to the correct solution.

Discussion. On the digits data sets, BA matches the performance of congealing. On the curves data sets, BA does substantially better than congealing in many cases, but in some cases congealing does slightly better. In all the experiments, both congealing and BA converged. BA's advantage is largely due to its explicit regularization of transformations which enables it to perform a maximization at each iteration. Congealings lack of such regularization requires it to take small steps at each iteration making it more susceptible to local optima. Furthermore, congealing typically requires five times the number of iterations to converge.

4.2 Clustering Extension using Bayesian Nonparametrics

We now extend the BA model introduced in the previous section to explicitly cluster the data points. This provides a mechanism for handling complex data sets that may contain multiple groups.

The major drawback of the BA model is that a single pair of data and transformation parameters (θ and φ, respectively) generate the entire data set. One natural extension to this generative process is to assume that we have several such parameter pairs (finite but unknown a priori) and each data point samples its parameter pair. By virtue of points sampling the same parameter pair, they are assigned to the same group or cluster. A Dirichlet process (DP) provides precisely this construction and serves as the prior for the data and transformation parameter pairs.

A DP essentially provides a distribution over distributions, or, more formally, a distribution on random probability measures (for a great overview on Bayesian statistics and Dirichlet processes, we refer the reader to Chapter 2 of Erik Sudderth's PhD Thesis [89]). It is parameterized by a base measure and a concentration parameter. A draw from a DP generates a finite set of samples from the base measure (the concentration parameter controls the number of samples). A key advantage of DP's is that the number of unique parameters (i.e. clusters) can grow and adapt to each data set depending on its size and characteristics. Under this new probability model, data points are generated in the following way:

1. Sample from the DP, $G \sim DP(\gamma, H_\alpha \times H_\lambda)$. γ is the concentration parameter, and H_α and H_λ are the base measures for $F_T(\varphi)$ and $F_D(\theta)$ respectively.

2. For each data point, x_i, sample a data and transformation parameter pair, $(\theta_i, \varphi_i) \sim G$.

3. Sample a transformation and canonical data item from their distributions, $y_i \sim F_D(\theta_i)$ and $\rho_i \sim F_T(\varphi_i)$.

4. Transform the canonical data item to generate the observed sample, $x_i = \tau(y_i, \rho_i)$.

Figure 4.6 depicts the generative process as described above (distributional form, right) and in the more traditional graphical representation with the cluster random variable, z, and mixture weights, π, made explicit (left).

Our model can thus be seen as an extension of the standard Bayesian infinite mixture model where we introduced an additional latent variable, ρ_i, for each data point to represent its transformation. Several existing alignment models [21, 22, 49, 58] can be viewed as similar extensions to other standard generative models. Sometimes the transformations are applied to other model parameters instead of data points as in

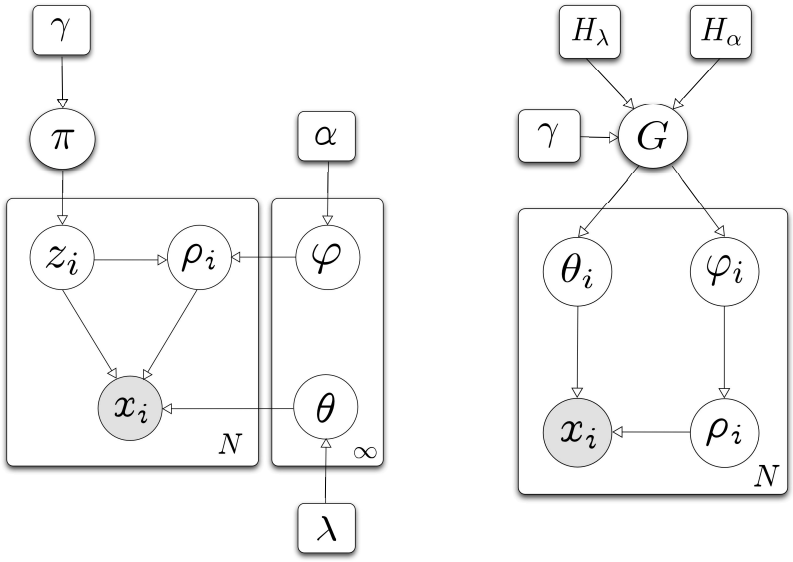

Figure 4.6: Graphical representation for our proposed nonparametric Bayesian joint alignment and clustering model (left) and its corresponding distributional form (right).

the case of transformed Dirichlet processes (TDP) [90]. TDP is an extension of *hierarchical Dirichlet processes* where global mixture components are transformed before being reused in each group. The challenge in introducing additional latent variables is in designing efficient learning schemes that can accommodate this increase in model complexity.

4.2.1 Learning

The model likelihood for N data points and K discovered clusters factorize according to the graph structure (Figure 4.6) in the following way:

$$
\begin{aligned}
p(\mathbf{x}, \boldsymbol{\rho}, \mathbf{z}, \boldsymbol{\theta}, \boldsymbol{\varphi}, \pi \mid \alpha, \lambda, \gamma) &= p(\pi|\gamma) \left(\prod_{k=1}^{K} p(\varphi_k|\lambda) p(\theta_k|\alpha) \right) \cdots \\
&\quad \left(\prod_{i=1}^{N} p(z_i|\pi) p(\rho_i|z_i, \boldsymbol{\varphi}) p(x_i|z_i, \rho_i, \boldsymbol{\theta}) \right) \\
&= p(\pi|\gamma) \left(\prod_{k=1}^{K} p(\varphi_k|\lambda) p(\theta_k|\alpha) \right) \cdots \\
&\quad \left(\prod_{i=1}^{N} p(z_i|\pi) p(\rho_i|\varphi_{z_i}) p(y_i|\theta_{z_i}) \right).
\end{aligned}
$$

We consider two different learning schemes for this model. The **first** is a blocked, Rao-Blackwellized Gibbs sampler, where we sample both the cluster assignment z_i, and transformation parameters ρ_i, simultaneously:

$$
\begin{aligned}
(z_i^{(t)}, \rho_i^{(t)}) &\sim p(z_i, \rho_i | \mathbf{z}_{-i}^{(t)}, \boldsymbol{\rho}_{-i}^{(t)}, \mathbf{x}, \gamma, \alpha, \lambda) \\
&\propto p(z_i | \mathbf{z}_{-i}^{(t)}, \gamma) p(\rho_i | \boldsymbol{\rho}_{-i}^{(t)}, \alpha) p(y_i | \mathbf{y}_{-i}^{(t)}, \lambda).
\end{aligned}
$$

As with the BA model, we approximate $p(\rho_i|\boldsymbol{\rho}_{-i}^{(t)}, \alpha) p(y_i|\mathbf{y}_{-i}^{(t)}, \lambda)$ with a point estimate based on its mode. Consequently this learning scheme is a direct generalization of the one derived for the BA model. Note that $p(z_i|\mathbf{z}_{-i}^{(t)}, \gamma)$ is the cluster predictive distribution based on the Chinese restaurant process (CRP) [6].

While this sampler is effective (it produced the positive result in Figure 4.1) it scales linearly with the number of clusters and computing the most likely transformation for a cluster is an expensive operation. We designed an alternative sampling scheme that does not require the expensive mode computation and whose running time is independent of the number of clusters.

The **second** sampler further integrates out the transformation parameter, and only samples the cluster assignment. We now derive an implementation for this sampler.

$$
\begin{aligned}
\forall_{i=1:N}\ z_i^{(t)} &\sim p(z_i \mid \mathbf{z}_{-i}^{(t)}, \mathbf{x}, \gamma, \alpha, \lambda) \\
&\propto p(z_i, x_i \mid \mathbf{z}_{-i}^{(t)}, \mathbf{x}_{-i}, \gamma, \alpha, \lambda) \\
&= p(z_i \mid \mathbf{z}_{-i}^{(t)}, \gamma) p(x_i \mid \mathbf{z}^{(t)}, \mathbf{x}_{-i}, \alpha, \lambda).
\end{aligned}
$$

$$
\begin{aligned}
p(x_i \mid \mathbf{z}, \mathbf{x}_{-i}, \alpha, \lambda) &= \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} \int_{\rho_i} p(x_i, \rho_i, \boldsymbol{\theta}, \boldsymbol{\varphi} \mid \mathbf{z}, \mathbf{x}_{-i}, \alpha, \lambda)\ d\rho_i\ d\boldsymbol{\varphi}\ d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} \left(\int_{\rho_i} p(x_i, \rho_i \mid z_i, \boldsymbol{\theta}, \boldsymbol{\varphi}, \alpha, \lambda)\ d\rho_i \right) p(\boldsymbol{\theta}, \boldsymbol{\varphi} \mid \mathbf{z}_{-i}, \mathbf{x}_{-i}, \alpha, \lambda)\ d\boldsymbol{\varphi}\ d\boldsymbol{\theta} \\
&\overset{(1)}{\approx} \int_{\rho_i} p(x_i, \rho_i \mid z_i, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\varphi}}, \alpha, \lambda)\ d\rho_i, \\
&\quad s.t.\ (\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\varphi}}) = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\varphi}} p(\boldsymbol{\theta}, \boldsymbol{\varphi} \mid \mathbf{z}_{-i}, \mathbf{x}_{-i}, \alpha, \lambda) \\
&= \int_{\rho_i} p(\rho_i \mid \hat{\boldsymbol{\varphi}}, z_i, \alpha) p(x_i \mid \rho_i, z_i, \hat{\boldsymbol{\theta}}, \lambda)\ d\rho_i \\
&= \int_{\rho_i} p(\rho_i \mid \hat{\varphi}_{z_i}, \alpha) p(x_i \mid \rho_i, \hat{\theta}_{z_i}, \lambda)\ d\rho_i \\
&\approx \frac{1}{L} \sum_{l=1}^{L} p(x_i \mid \hat{\rho_i}^l, \hat{\theta}_{z_i}, \lambda) \\
&\quad s.t.\ \{\hat{\rho_i}^l\}_{l=1}^{L} \sim p(\rho_i \mid \hat{\varphi}_{z_i}, \alpha) \\
&\overset{(2)}{\approx} \frac{\sum_{l=1}^{L} w_i^l \cdot p(x_i \mid \hat{\rho}^l, \hat{\theta}_{z_i}, \lambda)}{\sum_{l=1}^{L} w_i^l} \\
&\quad s.t.\ \{\hat{\rho}^l\}_{l=1}^{L} \sim q(\rho), \quad w_i^l = \frac{p(\hat{\rho}^l \mid \hat{\varphi}_{z_i}, \alpha)}{q(\hat{\rho_i}^l)}
\end{aligned}
$$

(1) approximates the posterior distribution of the parameters by its mode. The mode is computed using incremental hard-EM. Furthermore, the mode can be computed for each cluster's parameters independently. For every other data point j, perform an EM update:

$$
\begin{aligned}
\text{E:} \quad \hat{\rho}_j &= \arg\max_{\rho_j} p(\rho_j \mid x_j, \hat{\theta}_{z_j}, \hat{\varphi}_{z_j}) \\
&= \arg\max_{\rho_j} p(\rho_j \mid \hat{\varphi}_{z_j}) p(x_j \mid \rho_j, \hat{\theta}_{z_j})
\end{aligned}
$$

$$\text{M:} \quad \hat{\theta}_{z_j} = \arg\max_{\theta} p(\theta, \,|\, \{x_k, \rho_k \,|\, z_k = z_j\}, \lambda)$$
$$\hat{\varphi}_{z_j} = \arg\max_{\varphi} p(\varphi \,|\, \{\rho_k \,|\, z_k = z_j\}, \alpha)$$

(2) uses importance sampling in order to reduce the number of data transformations that need to be performed. Computing $p(x_i \,|\, \hat{\rho_i}^l, \hat{\theta}_{z_i}, \lambda)$ requires transforming the data point, which is the most computationally expensive single operation for this sampler. Thus it would be wise to reuse the samples, $\{\hat{\rho_i}^l\}_{l=1}^L$, across different clusters. We achieve this through importance sampling, which proceeds by sampling a set of transformation parameters from a proposal distribution, $q(\rho)$ and using those samples for all the clusters by reweighting them differently for each cluster. This is a large computational saving since the number of data transformation operations performed in a single iteration of this sampler is now independent of the number of clusters. Furthermore, the quality of approximation is controlled by the number of samples, L, generated.

To further increase the efficiency of the sampler, we approximate the maximization in the E-step by reusing the samples and selecting the one that maximizes $p(\rho_j \,|\, x_j, \hat{\theta}_{z_j}, \hat{\varphi}_{z_j})$. This avoids the direct maximization operation in the E-step which can be expensive. While not adopted in this work, further computational gains might be achieved at the expense of memory by storing and reusing samples (i.e. transformed data points) across iterations and reweighting them accordingly.

Thus our sampler iterates over every point in the data set, samples a cluster assignment and then updates $\hat{\theta}$ and $\hat{\varphi}$ for the sampled cluster. It also updates its own transformation parameter, $\hat{\rho}_i$ in the process.

Summary. We presented two samplers for our joint alignment and clustering model. Both samplers work well in practice, but the second is more efficient. For both samplers, every iteration begins by randomly permuting the order of the points and the DP concentration parameter is resampled using auxiliary variable methods [18].

As in the BA model, we cache the sufficient statistics for every cluster which can be updated efficiently as points are reassigned to clusters to allow for efficient likelihood and mode computation.

4.2.2 Incorporating Labelled Examples

The model presented in the previous section was used without any supervision. Supervision here refers to the ground-truth labels for some of the data points or the correct number of clusters. However, there are many scenarios where this information is available and would be advantageous to incorporate.

It is straightforward to modify the joint alignment and clustering model to accommodate such labelled examples. Lets assume we have positive examples for each cluster as well as a large data set of unlabeled examples. Before attempting to align and cluster the unlabeled examples, we would initialize several clusters and assign the positive examples to their respective clusters. By assigning these examples to their clusters and updating the sufficient statistics accordingly, the cluster parameters have incorporated the positive examples. Depending on the strength of the priors (i.e. the hyperparameters) and the number of positive examples per cluster it may be necessary to add the positive examples several times. The stronger the prior, the more times the positive examples need to be replicated. Note that replicating the positive examples does not increase memory usage since we only store the sufficient statistics for each cluster.

If the labelled portion contains positive examples for all the clusters, then setting the concentration parameter of the DP to 0 would prevent additional, potentially unnecessary, clusters from being created.

4.2.3 Experiment: Alignment and Clustering of Digits

We evaluated our unsupervised and semi-supervised models on two challenging data sets. The first contains 100 images of the digits "4" and "9" (Figure 4.1), which

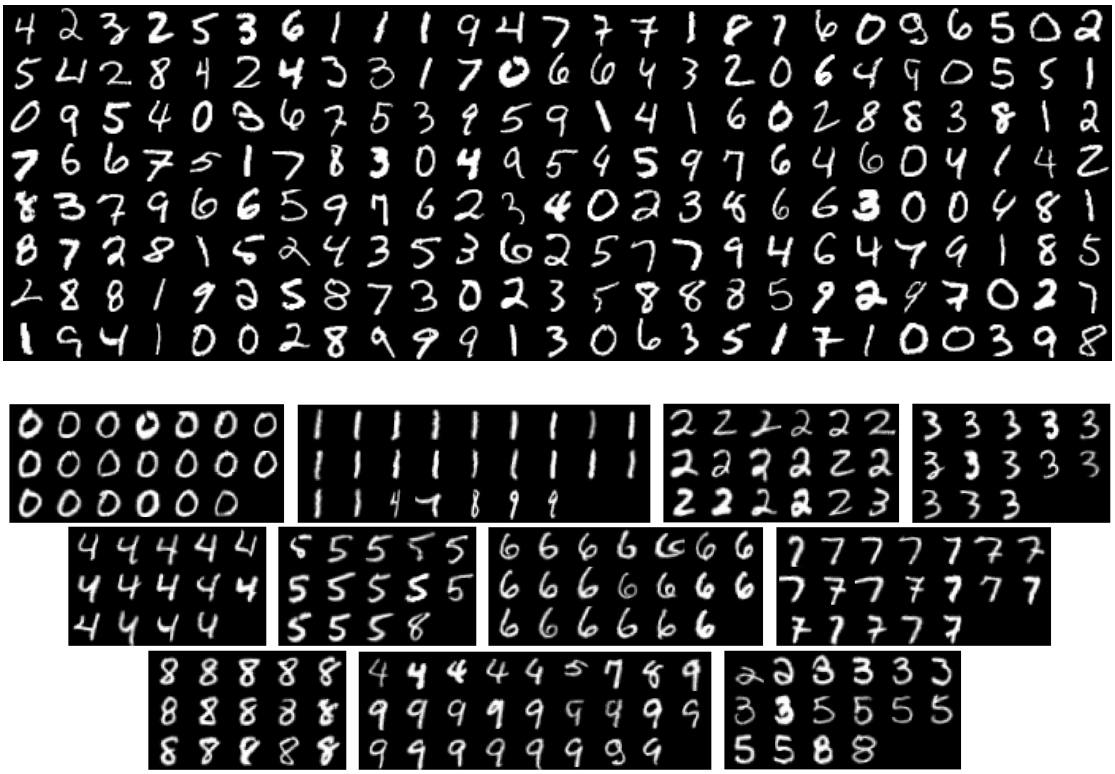

Figure 4.7: Unsupervised joint alignment and clustering of 200 images of all 10 digits. (Top) All 200 images provided to our model. (Bottom) The 11 clusters discovered and their alignments.

are the two most similar and confusing digit classes (the performance of KMeans on this data set is close to random guessing). The second contains the 200 images of all 10 digit classes used by Liu *et al.* [59] (Figure 4.7).[4] For the second data set we used the histogram of oriented gradients (HOG) feature representation [15] used by Liu *et al.* to enable a fair comparison.

For both digit data sets we compared several algorithms using the same two metrics reported by Lui *et al.*: *alignment score* measures the distance between pairs of aligned images assigned to the same cluster (we report the mean and standard de-

[4] Liu *et al.* also evaluated their model on 6 Caltech-256 categories and the CEAS face data set. For both data sets they randomly selected 20 images from each category. We found that the difficulty of a data set varied greatly from one sample to another, so we reached out to the authors. Unfortunately, they were only able to provide us with the digits data set which we do use. The digits data set was the most difficult of the three.

Algorithm	Digits 4 and 9 (Fig 4.1)	
	Alignment	Clustering
Original	4.54 (1.69) ± 0.034	–
KMeans	4.18 (1.57) ± 0.031	54.0%
Affinity propagation [20]	4.30 (1.77) ± 0.043	82.0%, 3
Infinite mixture model [18]	3.64 (1.34) ± 0.036	86.0%, 4
Congealing [49]	1.80 (0.98) ± 0.020	90.0%
Unsupervised JAC [§ 4.2.1]	1.47 (0.71) ± 0.014 **1.20 (0.56) ± 0.011**	**94.0**%, 2
Semi-supervised JAC [§ 4.2.2]	1.69 (0.84) ± 0.017 1.26 (0.63) ± 0.013	97.0%

Algorithm	All 10 digits (Fig 4.7)	
	Alignment	Clustering
Original	5.22 (1.86) ± 0.043	–
KMeans	4.88 (1.61) ± 0.033	62.5%
Affinity propagation [20]	4.74 (1.59) ± 0.040	67.5%, 14
Infinite mixture model [18]	4.86 (1.62) ± 0.036	69.0%, 13
Congealing [49]	3.43 (1.33) ± 0.029	64.0%
TIC [21]	6.00 (1.1)	35.5%
Unsupervised SAC [59]	3.80 (0.9)	56.5%
Semi-supervised SAC [59]	not reported	73.7%
Unsupervised JAC [§ 4.2.1]	2.56 (1.27) ± 0.030 **1.58 (0.94) ± 0.022**	**87.0**%, 11
Semi-supervised JAC [§ 4.2.2]	2.87 (1.34) ± 0.030 1.80 (1.03) ± 0.023	84.5%

Table 4.2: Joint alignment and clustering of images. The top subtable refers to the first digit data set comprising 100 images of the digits "4" and "9" (Figure 4.1), while the bottom subtable refers to the second data set comprising 200 images of all 10 digits (the same data set used by Liu *et al.* [59], Figure 4.7). The alignment score columns contain three metrics that adhere to the following template: mean (standard deviation) ± standard error. The number following the clustering accuracy in the "Affinity propagation", "Infinite mixture model" and "Unsupervised JAC" rows is the number of clusters that the model discovered (i.e. chose to represent the data with). For both unsupervised and semi-supervised JAC we include the alignment score from the Gibbs sampler and after the cluster-specific post-processing (see text for more details). On both data sets, our models significantly outperform previous nonparametric alignment [49], clustering [20, 18], and joint alignment and clustering [21, 59].

viation of all the distances, and the standard error[5]), and *clustering accuracy* is the Rand index with respect to the correct labels. Table 4.2 summarizes the results on the models we evaluated:

- Original: we computed the alignment score assuming correct clustering to offer reference for what the alignment score would be without transforming the data.
- KMeans: we clustered the digits into the correct number of ground-truth classes using the best of 200 KMeans runs.
- Affinity propagation: which serves as an effective clustering baseline that can also recover the number of clusters automatically.
- Infinite mixture model: removing the transformation/alignment component of our model reduces it to a standard Bayesian infinite mixture model. We ran this model to evaluate the advantage of joint alignment and clustering.
- Congealing: we ran congealing on all the images simultaneously and after alignment converged, clustered the aligned images using KMeans (with the correct number of ground-truth clusters). This allows us to evaluate the advantages of simultaneous alignment and clustering over alignment followed by clustering. [6]
- TIC, USAC and SSAC results are listed exactly as reported by Liu *et al*.
- Unsupervised JAC refers to our full nonparametric Bayesian alignment and clustering model (§ 4.2.1).

---

[5] The standard error here is defined as the sample standard deviation divided by the square root of the number of pairs.

[6] The congealing results on the data set of "4" and "9" (Table 4.2) are significantly better than what was included in our 2012 UAI paper. The reason for this is that we switched from using the binary congealing implementation provided online to our own (overviewed in Chapter 5), which has a more effective optimization procedure.

- Semi-supervised JAC refers to the semi-supervised variant of our alignment and clustering model (§ 4.2.2). We used a *single* positive example for each digit and set the DP concentration parameter to 0.

Note that the alignment scores for KMeans and the infinite mixture model are not relevant since no alignment takes place in either of these two algorithms. They are only included to offer a reference for the alignment score when the data is not transformed.

Both the unsupervised and semi-supervised joint alignment and clustering models have two alignment scores. The first is the alignment score after transforming the instances with the mean transformation parameters generated from the Gibbs sampler. However, this can be a slightly noisy estimate (especially if the chain is not long enough). Consequently, we perform cluster-specific alignment using the Bayesian alignment model (initializing the transformation parameters to those produced by the joint alignment and clustering model). This step always improves alignment quality and is included in Tables 4.2 and 4.3 as the second alignment score.

As the results show, our models outperform previous work with respect to both alignment and clustering quality. We make three observations about these results:

1. Our unsupervised model outperformed the unsupervised model of Liu *et al.* by 30.5%, and our semi-supervised model outperformed their semi-supervised model by 10.8%. This is in addition to the improvement in alignment quality.

2. Our unsupervised model improved upon the standard infinite mixture model in terms of alignment quality, clustering accuracy, and correctness of the discovered number of clusters.

3. The number of clusters discovered by our unsupervised model is quite accurate. For the first data set the model discovered the correct number of clusters (see Figure 4.1), and for the second it needed one additional clusters (see Figure 4.7).

Algorithm	ECG Heart Beats (Fig 4.8)	
	Alignment	Clustering
KMeans (2 clusters)	$75.92\ (57.88) \pm 1.923$	58.70%
KMeans (5 clusters)	$40.98\ (28.54) \pm 1.753$	82.61%
Affinity propagation [20]	$54.32\ (36.80) \pm 1.871$	$65.22\%, 4$
Infinite mixture model [18]	$48.73(45.80) \pm 2.956$	$78.26\%, 2$
Congealing (entropy, 5 clusters) [49]	$56.19\ (38.97) \pm 2.080$	69.57%
Congealing (variance, 5 clusters) [49]	$53.85\ (37.37) \pm 2.136$	76.09%
Unsupervised JAC [§ 4.2.1]	$10.48\ (10.22) \pm 0.591$ $\mathbf{5.93}\ (\mathbf{4.83}) \pm \mathbf{0.279}$	$\mathbf{86.96}\%, 5$

Table 4.3: Joint alignment and clustering of ECG data. The alignment score columns contain three metrics that adhere to the following template: mean (standard deviation) ± standard error. The number following the clustering accuracy in the "Affinity propagation", "Infinite mixture model" and "Unsupervised JAC" rows is the number of clusters that the model discovered (i.e. chose to represent the data with).

These positive results validate our joint alignment and clustering models and associated learning schemes. Furthermore, it provides evidence for the advantage of solving both alignment and clustering problems simultaneously.

4.2.4 Experiment: Alignment and Clustering of Curves

We now present joint alignment and clustering results on a challenging curve data set of ECG heart data [45] that is helpful in identifying the heart condition of patients. This data set contains 46 curves. 24 represent a normal heartbeat, and 22 represent an abnormal heartbeat. We ran both congealing and our nonparametric Bayesian joint alignment and clustering model. In both cases we excluded the non-linear scaling in amplitude transformation since the amplitudes of the curves are helpful in classifying whether the curve is normal or abnormal.

Figure 4.8 shows the result of our unsupervised model as well as congealing with both entropy and variance, while Table 4.3 presents quantitative alignment and clustering metrics for several techniques. Our model discovered 5 clusters resulting in a clustering accuracy of 86.96%. Inspecting the clusters discovered by our model

highlights the fact that although the data set represents two groups (normal and abnormal), the curves do not naturally fall into two clusters and more are needed to explain the data appropriately. This validates the utility in discovering clusters in a data-driven manner. Overall, our model provides a large improvement in both alignment and clustering quality compared to prior work.

4.3 Large-scale Learning

In this section we discuss the adaptation of our joint alignment and clustering model to both online (when the data arrives at intervals) and distributed (when multiple processors are available) settings. Both of these adaptations are applicable to the unsupervised and semi-supervised settings.

4.3.1 Online Learning

There are several scenarios where online alignment and clustering may be helpful. Consider for instance a very large data set that cannot fit in memory or the case where the data set is not available up front but arrives over an extended period of time (such as in a tracking application).

An advantage of our model that has not yet been raised is its ability to easily adapt to an online setting where only a portion of the data set is available in the beginning. This is due to our use of conjugate priors and distributions in the exponential family which enable us to efficiently summarize an entire cluster through its sufficient statistics. Consequently, we can align/cluster the initial portion of the data set and save out the sufficient statistics for every cluster after each iteration (for both the data and transformations). Then as new data arrives, we can load in the sufficient statistics and use them to guide the alignment and clustering of the new data in lieu of the original data set which can now be discarded.

Given a sufficiently large initial data set, the alignment of a new data point using the procedure described above would be nearly identical to the result had that data point been included in the original set. This is true since the addition of a single point to an already large data set would have a negligible effect on the sufficient statistics. This process is also applicable to the Bayesian alignment model.

4.3.2 Distributed Learning

We now describe how to adapt our sampling scheme to a distributed setting using the MapReduce [16] framework. This facilitates scaling our model to large data sets in the presence of many processors. The key difference between the MapReduce implementation and the one described in § 4.2.1 is that the cluster parameters are updated once per sampling iteration instead of after each point's reassignment (i.e. using a standard sampler instead of a Rao-Blackwellized sampler).

A MapReduce framework involves two key steps, Map and Reduce. For our model the mapper would handle updating the transformation parameter and clustering assignment of a single data point, while the reducer would handle updating the parameters of a single cluster. More specifically, the input to each Map operation would be a data point along with a snapshot of the model parameters (the set of sufficient statistics that summarize the data set). The Map would output the updated cluster assignment and transformation parameter for that data point. The input to the Reduce step would then be all the data points that were assigned to a specific cluster (i.e. we would have a Reduce operation for every cluster created). The Reducer would then update the cluster parameters. Thus each sampling iteration is composed of a Map and Reduce stage.

4.4 Summary

We presented a Bayesian alignment model that was more effective than congealing on unimodal curve and image data sets. We extended this model to include clustering and showed its success on complex digit data sets and ECG heart signals. The model outperforms many alignment-only, clustering-only and previous joint alignment and clustering algorithms. We also presented different variations of our model including semi-supervised, online and distributed.

A strength of our model is the separation of the transformation function and sampling scheme, which makes it applicable to a wide range of data types, feature representations, and transformation functions. In this chapter we presented results on three data types (2D points, 1D curves, and images), three transformation functions (point rotations, nonlinear curve transformations and affine image transformations), and two feature representations (identity and HOG).

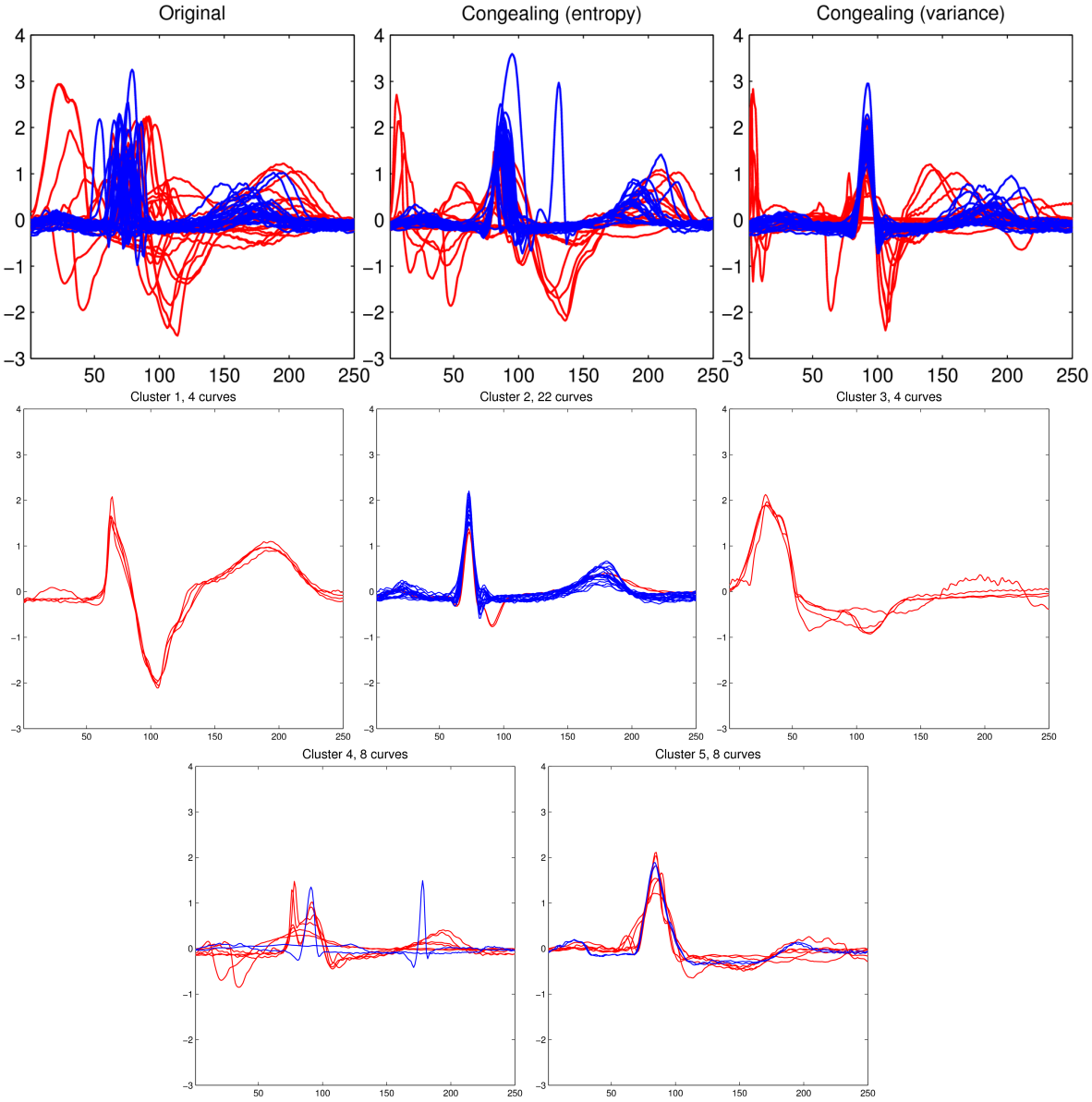

Figure 4.8: Joint alignment and clustering of ECG heart data (two classes, color-coded: red for abnormal, and blue for normal). The **first row** displays the original data set (left) and the result of congealing with entropy (left) and congealing with variance (right). The **last two rows** display the 5 clusters discovered by our unsupervised model.

CHAPTER 5

SOFTWARE AND REPRODUCIBLE RESEARCH

Associated with this thesis is a software package written in Matlab that will be made available for download and contributions on GitHub. This software package will contain implementations for all the models introduced in the thesis as well as common algorithms that were either used or compared against such as congealing, infinite mixture models, and learning convolutional restricted Boltzmann machines. We hope that it will serve three purposes:

- Expose implementation details, as well as enable the reproduction of the results presented in this thesis.
- Facilitate the process of comparing against our model on other data sets.
- Encourage improvements, contributions, and extensions to our work.

Consequently, the package is cleanly organized into several classes defined by abstract classes that enable the easy addition of new data types, transformation functions or feature representations. Figure 5.1 contains a schematic showing the base classes in our library [1], while Figures 5.2, 5.3, 5.4, and 5.5 show how these base classes are extended and used by more complex algorithms. In all these figures, interfaces and abstract classes are represented by rectangular boxes, while concrete classes are represented by rounded boxes. Furthermore, dashed arrows represent

[1] Class names currents share a common "vis" prefix, although that might change in the public code release.

composition (i.e. the class at the head of the arrow uses an instance of a class at the tail of the arrow), while solid arrows represent inheritance (i.e. the class at the head of the arrow extends the class at the tail of the arrow).

We now overview the base classes listed in Figure 5.1:

- **visDataSet:** is an interface that encapsulates a data set or ensemble. It has two concrete classes, visImageDataSet which stores a collection of images, and visVectorDataSet which stores any data type that can be represented as a vector (e.g. 1D curves).

- **visTransforms:** is an interface that encapsulates transformation functions and specifies a mechanism for defining them. It has three concrete classes, visCurveTransforms which implements the curve transformation function outlined in Chapter 2 (which employs nonlinear time scaling, nonlinear amplitude scaling, linear amplitude scaling and amplitude translation); visImageTransforms which implements the 7 affine image transformations (x-y translation, scaling, and shearing and rotations); and visRotatePointTransforms which implements rotating a 2D point around the origin (used in our illustrative example in Figure 4.2).

- **visProcessor:** is an interface that encapsulates computing a feature representation of the data. The default behavior is to return the data as is. It has three concrete classes, visDiscretize which simply discretizes the data into a predetermined number of bins; visHOG which computes the histogram of oriented gradients feature representation used in Chapter 4 for the joint alignment and clustering of digits; and visCDBN which learns a convolutional deep Boltzmann machines and returns the pooling unit activations as a feature representation (used for the alignment and classification of curves in Chapter 3).

- **visSufficientStatistics:** is an interface that encapsulates the sufficient statistics for a random variable belonging to the exponential family. It defines the interface for updating the sufficient statistics, and computing the posterior mode and likelihoods assuming conjugate priors. These sufficient statistics are used to represent the data and/or transformation parameters for the Bayesian alignment, infinite mixture model and joint alignment and clustering models. It has two concrete classes, visGaussianSufficientStatistics which implements the sufficient statistics for a Gaussian random variable with a Normal-inverse-Wishart prior (including diagonal or full covariance matrices, as well as fixed mean or covariances); and visMultinomialSufficientStatistics which implements the sufficient statistics for a multinomial random variable with a Dirichlet prior.

- **visAlignment:** is an abstract class that encapsulates alignment algorithms. Both congealing and Bayesian alignment are direct implementations of this abstract class.

- **visMixtureModel:** is an abstract class that encapsulates clustering based on finite mixture models using a Rao-Blackwellized (or collapsed) Gibbs sampler. visDPClustering is a direct implementation of this abstract class.

It should be straighforward to see how the design of the software package takes advantage of the fact that both congealing and the models presented in the thesis are frameworks and can be applied to any data type, feature representation or transformation function. We hope that by making our implementations publicly available we enable the application of our models to a wide range of data sets. We also hope to encourage other researchers to contribute their algorithms to this software package so that it evolves into a collection of well-maintained tools for alignment and joint alignment and clustering of complex data sets.

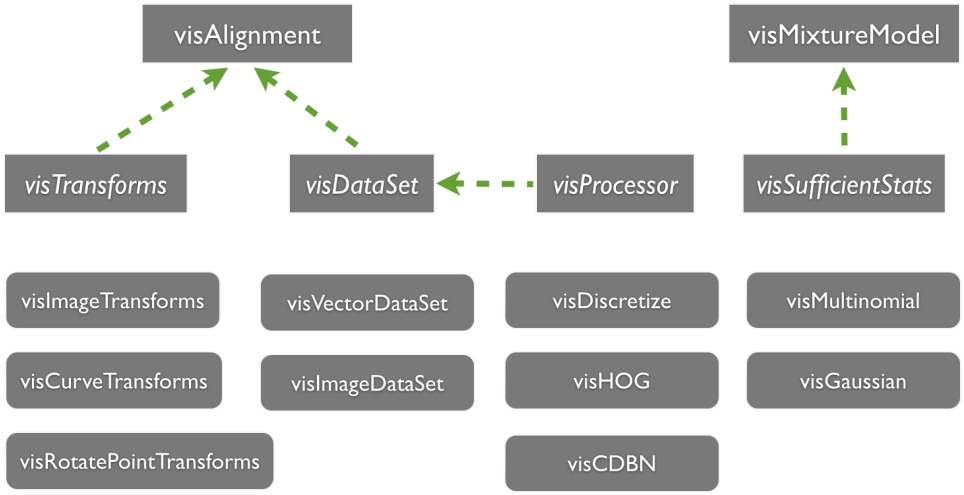

Figure 5.1: Layout of the base classes included in our software package.

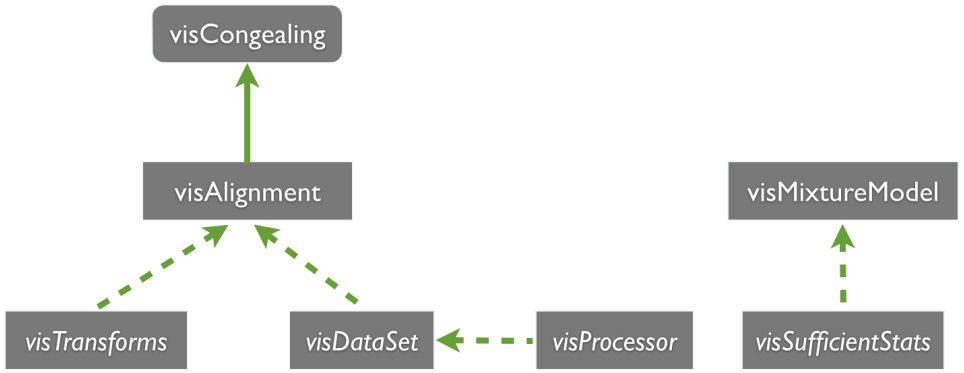

Figure 5.2: The visCongealing class extends the visAlignment abstract class.

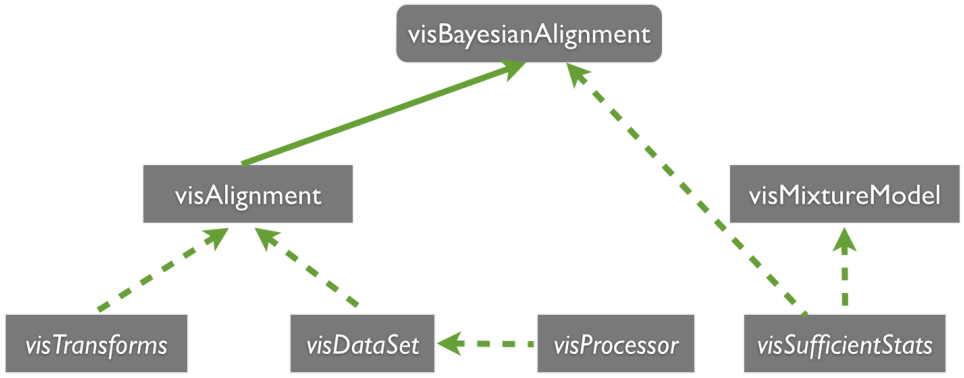

Figure 5.3: The visBayesianAlignment class extends the visAlignment abstract class and uses two visSufficientStats objects to represent the data being aligned and the transformation parameters.

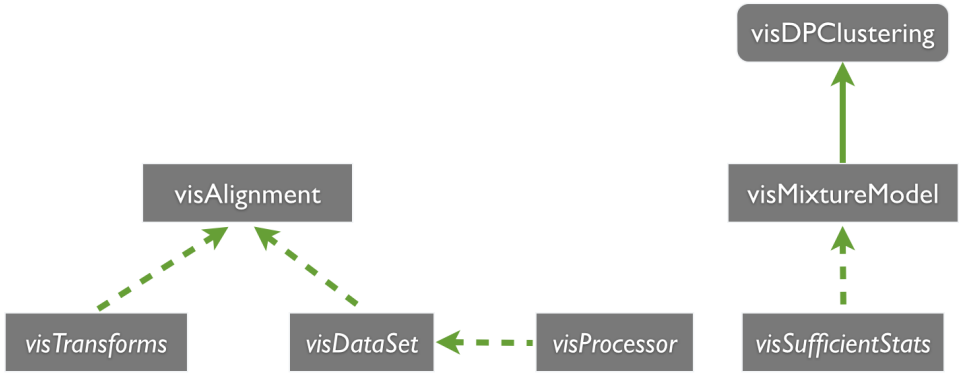

Figure 5.4: The visDPClustering class (which implements infinite mixture models) extends the visMixtureModel class to allow the creation and removal of classes within the sampling scheme.

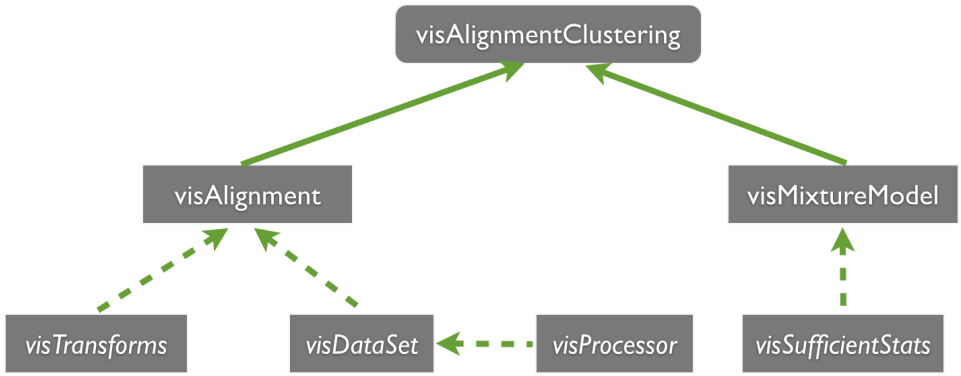

Figure 5.5: The visAlignmentClustering extends both the visAlignment and visMixtureModel classes to allow the alignment and clustering of instances. Matlab allows multiple-inheritance.

CHAPTER 6

CONCLUSION AND SUMMARY

The focus of this thesis is on designing probabilistic alignment algorithms that can handle complex data sets. We introduced a congealing algorithm for 1-dimensional curves that uses a flexible and efficient set of curve parameterizations. We showed how joint alignment of curve data can be a useful mechanism for generating different views of a data set that can aid in classification. We improved upon the congealing framework by incorporating features based on convolutional restricted Boltzmann machines. This resulted in a powerful curve representation that improved both alignment and classification. Finally, we presented a nonparametric Bayesian joint alignment and clustering model that is scalable and applicable to different data types. The model outperforms prior alignment-only, clustering-only and joint alignment and clustering models, and has semi-supervised, distributed and online variations that can be useful in a variety of scenarios. We also presented an open-source software library that allows others to replicate our experiments and utilize our models with minimal effort.

Overall, this thesis takes steps towards developing an unsupervised data processing pipeline that includes alignment, clustering and feature learning. While clustering and feature learning serve as auxiliary information to improve alignment, they are important byproducts. This opens the door for several interesting future directions and applications:

- Closing the loop on unsupervised learning: in this thesis we showed how unsupervised feature learning can aid alignment and how clustering and alignment are interrelated processes that ought to be solved together. Ideally, we would

also incorporate unsupervised feature learning into our joint alignment and clustering model, where the features are also updated throughout the alignment (and even the clustering if we choose to learn cluster-specific features). Updating the features would help us discover better clusters and improve alignment which should in turn further improve the feature representation. Performing all three unsupervised processes simultaneously would provide a powerful data processing module that can be useful for various learning tasks.

- While we presented a MapReduce implementation of our joint alignment and clustering model that helps it scale to a distributed environment, exploring alternative parameter learning schemes based on variational inference [7, 48, 24, 56] would improve its efficiency for single-core machines. This would be particularly advantageous if we incorporated feature learning as outlined above.

- Incorporate transformation-specific kernels for both alignment and clustering, and classification based on multiple kernel learning. When classifying curves using transformation parameters in Chapters 2 and 3 or jointly aligning and clustering a data set in Chapter 4, we used a naive representation of the transformation parameters. In the future, we would like to explore utilizing transformation-specific kernels (e.g. [65]) that take into account the topology of the transformation operator.

Finally, we would like to explore alternative application areas that can benefit from the models introduced in this thesis (and the proposed extensions above). For example, our curve congealing algorithm can be used to align camera trajectories [47] or event-related potential signals [62], and our joint alignment and clustering model can be used to group the motion trajectories of the elderly around their homes [98].

BIBLIOGRAPHY

[1] Aach, J, and Church, G M. Aligning gene expression time series with time warping algorithms. *Bioinformatics 17*, 6 (2001), 495–508.

[2] Ahammad, Parvez, Harmon, Cyrus, Hammonds, Ann, Sastry, Shankar, and Rubin, Gerald. Joint nonparametric alignment for analyzing spatial gene expression patterns of Drosophila imaginal discs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2005).

[3] Beirlant, J, Dudewicz, E J, Gyorfi, L, and van der Meulen, E C. Nonparametric entropy estimation: an overview. *International Journal of Mathematical and Statistical Sciences* (1997), 17–39.

[4] Bengio, Yoshua. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* (Dec. 2009).

[5] Bengio, Yoshua, Lamblin, Pascal, Popovici, Dan, and Larochelle, Hugo. Greedy Layer-Wise Training of Deep Networks. In *Advances in Neural Information Processing Systems* (2007).

[6] Blackwell, David, and MacQueen, James. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics 1* (1973), 353–355.

[7] Blei, David M, and Jordan, Michael I. Variational inference for Dirichlet process mixtures. *Journal of Bayesian Analysis 1*, 1 (2006), 121–144.

[8] Boureau, Y-Lan, Bach, Francis R, LeCun, Yann, and Ponce, Jean. Learning Mid-Level Features For Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[9] Cassella, George, and Robert, Christian P. Rao-Blackwellization of sampling schemes. *Biometrika 83*, 1 (1996), 81–94.

[10] Coates, Adam. *Demystifying Unsupervised Feature Learning.* PhD thesis, Stanford University, Dec. 2012.

[11] Collobert, Ronan, and Weston, Jason. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the International Conference on Machine Learning* (2008).

[12] Cootes, T F, Edwards, G J, and Taylor, C J. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2003).

[13] Cortes, Corinna, Mohri, Mehryar, and Rostamizadeh, Afshin. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems* (Dec. 2010).

[14] Cox, Mark, Lucey, Simon, Cohn, Jeffrey, and Sridharan, Sridha. Least squares congealing for unsupervised alignment of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[15] Dalal, Navneet, and Triggs, Bill. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2005).

[16] Dean, Jeffrey, and Ghemawat, Sanjay. MapReduce: Simplified data processing on large clusters. In *Symposium on Operating System Design and Implementation* (2004).

[17] Elad, M, Goldenberg, R, and Kimmel, R. Low bit-rate compression of facial images. *IEEE Transactions on Image Analysis 9*, 16 (2007), 2379–2383.

[18] Escobar, Michael D, and West, Mike. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association 90*, 430 (June 1995), 577–588.

[19] Fischer, Asja, and Igel, Christian. An Introduction to Restricted Boltzmann Machines. In *Proceedings of the 17th Iberoamerican Congress on Pattern Recognition* (Dec. 2012).

[20] Frey, Brendan J, and Dueck, Delbert. Clustering by Passing Messages between Data Points. *Science 315* (Feb. 2007).

[21] Frey, Brendan J, and Jojic, Nebojsa. Transformation-invariant clustering and dimensionality reduction using EM. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 1 (Jan. 2003), 1–17.

[22] Gaffney, Scott, and Smyth, Padhraic. Joint probabilistic curve clustering and alignment. In *Advances in Neural Information Processing Systems* (2004).

[23] Gelman, Andrew, Carlin, John B, Stern, Hal S, and Rubin, Donald B. Bayesian Data Analysis. Chapman and Hall, CRC Press, 2003.

[24] Gomes, Ryan, Welling, Max, and Perona, Pietro. Incremental learning of non-parametric Bayesian mixture models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[25] Gonen, Mehmet, and Alpaydin, Ethem. Multiple Learning Kernel Algorithms. *Journal of Machine Learning Research 12* (Jan. 2011).

[26] Gong, Yunchao, and Lazebnik, Svetlana. Iterative Quantization: A Procrustean Approach to Learning Binary Codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2012).

[27] Goodfellow, Ian J, Le, Quoc V, Saxe, Andrew M, Lee, Honglak, and Ng, Andrew Y. Measuring invariances in deep networks. In *Advances in Neural Information Processing Systems* (2009).

[28] Gower, J C. Generalized Procrustes Analysis. *Psychometrika 40* (1975), 33–51.

[29] Gupta, L, Molfese, D L, Tammana, R, and Simos, P G. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering 43*, 4 (Apr. 1996), 348–356.

[30] Hinton, Geffrey E. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation 14*, 8 (2002), 1771–1800.

[31] Hinton, Geoffrey E, Osindero, Simon, and Teh, Yee Whye. A fast learning algorithm for deep belief nets. *Neural Computation 18*, 7 (2006), 1527–1554.

[32] Hofmann, Thomas, Scholkopf, Bernhard, and Smola, Alexander J. Kernel Methods in Machine Learning. *The Annals of Statistics 36* (Apr. 2008), 1171–1220.

[33] Huang, Gary B. *Weakly supervised learning for unconstrained face processing.* PhD thesis, University of Massachusetts Amherst, 2012.

[34] Huang, Gary B, Jain, Vidit, and Learned-Miller, Erik G. Unsupervised joint alignment of complex images. In *Proceedings of the International Conference on Computer Vision* (2007).

[35] Huang, Gary B, Lee, Honglak, and Learned-Miller, Erik G. Learning Hierarchical Representations for Face Verification with Convolutional Deep Belief Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2012).

[36] Huang, Gary B, Mattar, Marwan A, Berg, Tamara, and Learned-Miller, Erik G. Labeled Faces in the Wild: A database for studying face recognition in unconstrained environments. In *IEEE ECCV Workshop on Faces in Real-Life Images* (2008).

[37] Huang, Gary B, Mattar, Marwan A, Lee, Honglak, and Learned-Miller, Erik G. Deep congealing. In *Advances in Neural Information Processing Systems* (2012).

[38] Huang, Gary B, Ramesh, Manu, Berg, Tamara, and Learned-Miller, Erik G. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Tech. rep., University of Massachusetts, Amherst, Oct. 2007.

[39] Hurley, J R, and Cattel, R B. The Procrustes program: Producing direct rotation to test a hypothesized factor structure. *Behavioral Science 7* (1962), 258–262.

[40] Jarrett, K, Kavukcuoglu, K, Ranzato, M, and LeCun, Y. What is the best multi-stage architecture for object recognition? In *Proceedings of the International Conference on Computer Vision* (2009).

[41] Jaskowski, P, and Verleger, R. Amplitudes and latencies of single-trial ERP's estimated by a maximum-likelihood method. *IEEE Transactions on Biomedical Engineering 46*, 9 (1999), 987–993.

[42] Jojic, Nebojsa, and Frey, Brendan J. Learning flexible sprites in video layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2001).

[43] Kendall, David. Shape manifolds, Procrustean metrics and complex projective spaces. *Bulletin of the London Mathematical Society 16* (1984), 81–121.

[44] Keogh, Eamonn, Xi, Xiaopeng, Wei, Li, and Ratanamahatana, Chotirat Ann. UCR Time Series Repository, Feb. 2008.

[45] Kim, Seyoung, and Smyth, Padhraic. Segmental hidden Markov models with random effects for waveform modeling. *Journal of Machine Learning Research 7* (2006), 945–969.

[46] Kristof, Walter, and Wingersky, Bary. A generalization of the orthogonal Procrustes rotation procedure to more than two matrices. In *Annual Convention of the American Psychological Association* (1971).

[47] Kuo, Thomas, Sunderrajan, Santhoshkumar, and Manjunath, B S. Camera Alignment using Trajectory Intersections in Unsynchronized Videos. In *Proceedings of the International Conference on Computer Vision* (Jan. 2013).

[48] Kurihara, Kenichi, Welling, Max, and Teh, Yee Whye. Collapsed variational Dirichlet process mixture models. In *International Joint Conference on Artificial Intelligence* (2007).

[49] Learned-Miller, Erik G. Data-driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 2 (Feb. 2006), 236–250.

[50] Learned-Miller, Erik G, and Ahammad, Parvez. Joint MRI bias removal using entropy minimization across images. In *Advances in Neural Information Processing Systems* (2004).

[51] Learned-Miller, Erik G, and Jain, Vidit. Many heads are better than one: Jointly removing bias from Multiple MRIs using nonparametric maximum likelihood. In *Proceedings of Information Processing in Medical Imaging* (2005), pp. 615–626.

[52] Lee, H, Battle, A, Raina, R, and Ng, A Y. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems* (2007).

[53] Lee, Honglak, Grosse, Roger, Ranganath, Rajesh, and Ng, A Y. Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM 54*, 10 (2011), 95–103.

[54] Lee, Honglak, Grosse, Roger, Ranganath, Rajesh, and Ng, Andrew Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the International Conference on Machine Learning* (2009).

[55] Lee, Honglak, Largman, Yan, Pham, Peter, and Ng, Andrew Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems* (2009).

[56] Lin, Dahua. Online Learning of Nonparametric Mixture Models via Sequential Variational Approximation. In *Advances in Neural Information Processing Systems* (Jan. 2013).

[57] Lisin, Dima, Mattar, Marwan A, Blaschko, Matthew, Benfield, Mark C, and Learned-Miller, Erik G. Combining Local and Global Image Features for Object Class Recognition. In *IEEE CVPR Workshop on Learning in Computer Vision and Pattern Recognition* (2005).

[58] Listgarten, Jennifer, Neal, Radford M, Roweis, Sam T, and Emili, Andrew. Multiple alignment of continuous time series. In *Advances in Neural Information Processing Systems* (2004).

[59] Liu, Xiaoming, Tong, Yan, and Wheeler, Frederick W. Simultaneous alignment and clustering for an image ensemble. In *Proceedings of the International Conference on Computer Vision* (2009).

[60] Liu, Xiaoming, Tong, Yan, Wheeler, Frederick W, and Tu, Peter H. Facial contour labeling via congealing. In *Proceedings of the European Conference on Computer Vision* (2010).

[61] Lowe, David. Scale Invariant Feature Transform. *International Journal of Computer Vision* (2004).

[62] Luck, Steven. *An Introduction to the Event-Related Potential Technique*. Cognitive Neuroscience, 2005.

[63] Mattar, Marwan A, Hanson, Allen R, and Learned-Miller, Erik G. Unsupervised Joint Alignment and Clustering using Bayesian Nonparametrics. In *Uncertainty in Aritifical Intelligence* (2012).

[64] Mattar, Marwan A, Ross, Michael G, and Learned-Miller, Erik G. Nonparametric curve alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (2009).

[65] Miller, Erik G, and Chefd'hotel, Christophe. Practical Non-parametric Density Estimation on a Transformation Group for Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2003).

[66] Miller, Erik G, Matsakis, Nick, and Viola, Paul. Learning from one example through shared densities on transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2000).

[67] Mosier, C I. Determining a simple structure when loadings for certain tests are known. *Psychometrika 4* (1939), 149–162.

[68] Muller, Meinard. Information Retrieval for Music and Motion. Springer, Apr. 2007.

[69] Myers, C S, and Rabiner, L R. A comparative study of several dynamic time-warping algorithms for connected word recognition. *The Bell System Technical Journal* (Apr. 1981).

[70] Neal, Radford M. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman and Hall, CRC Press, May 2011.

[71] Neal, Radford M, and Hinton, Geoffrey E. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models* (1998), pp. 355–368.

[72] Nguyen, Hieu V, and Bai, Li. Cosine Similarity Metric Learning for Face Verification. In *Proceedings of the Asian Conference on Computer Vision* (2010).

[73] Olshausen, Bruno A, and Field, David J. Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature 381* (1996), 607–609.

[74] Rahimi, Ali, and Recht, Ben. Learning to transform time series with a few examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 10 (Oct. 2007), 1759–1775.

[75] Rakthanmanon, Thanawin, Campana, Bilson, Mueen, Abdullah, Batista, Gustavo, Westover, Brandon, Zhu, Qiang, Zakaria, Jesin, and Keogh, Eamonn. Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping. In *ACM SIG Conference on Knowledge Discovery and Data Mining* (Apr. 2012).

[76] Ramsay, J O. Estimating Smooth Monotone Functions. *Journal of Royal Statistical Society 60*, 2 (1998), 265–375.

[77] Ramsay, J O, and Li, Xiaochun. Curve Registration. *Journal of Royal Statistical Society* (1998), 351–363.

[78] Ramsay, J O, and Silverman, B W. *Functional Data Analysis.* Springer-Verlag, New York, 1997.

[79] Ranzato, Marc'Aurelio, Poultney, Christopher, Chopra, Sumit, and LeCun, Yann. Efficient Learning of Sparse Representations with an Energy-Based Model. In *Advances in Neural Information Processing Systems* (2006), pp. 1137–1144.

[80] Ratanamahatana, Chotirat Ann, and Keogh, Eamonn. Everything you know about Dynamic Time Warping is Wrong . In *KDD Workshop on Mining Temporal and Sequential Data* (Apr. 2004).

[81] Rath, Tony, and Manmatha, R. Word Image Matching Using Dynamic Time Warping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Apr. 2003).

[82] Roth, Stefan, and Black, Michael J. Fields of Experts. *International Journal of Computer Vision 82*, 2 (Apr. 2009), 205–229.

[83] Salakhutdinov, Ruslan, and Hinton, Geoffrey E. Semantic Hashing. *International Journal of Approximate Reasoning 50* (2009), 969–978.

[84] Salvador, Stan, and Chan, Philip. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In *KDD Workshop on Mining Temporal and Sequential Data* (Apr. 2004).

[85] Sankoff, David, and Kruskal, Joseph. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addision-Wesley, Apr. 1983.

[86] Sibson, Robin. Studies in the Robustness of Multidimensional Scaling: Procrustes Statistics. *Journal of the Royal Statistical Society, Series B 40* (1978), 234–238.

[87] Snelson, Edward, Rasmussen, Carl Edward, and Ghahramani, Zoubin. Warped Gaussian processes. In *Advances in Neural Information Processing Systems* (2004).

[88] Sohn, Kihyuk, Jung, Dae Yon, Lee, Honglak, and Hero, III, Alfred. Efficient Learning of Sparse, Distributed, Convolutional Feature Representations for Object Recognition. In *Proceedings of the International Conference on Computer Vision* (2011).

[89] Sudderth, Erik B. *Graphical Models for Visual Object Recognition and Tracking.* PhD thesis, Massachusetts Institute of Technology, May 2006.

[90] Sudderth, Erik B, Torralba, Antonio, Freeman, William T, and Willsky, Alan S. Describing visual scenes using transformed Dirichlet processes. In *Advances in Neural Information Processing Systems* (2005).

[91] Vasicek, O. A Test for Normality Based on Sample Entropy. *Journal of Royal Statistical Society 38*, 1 (1976), 54–59.

[92] Vedaldi, Andrea, Guidi, Gregorio, and Soatto, Stefano. Joint alignment up to (lossy) transformations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[93] Vedaldi, Andrea, and Soatto, Stefano. A complexity-distortion approach to joint pattern alignment. In *Advances in Neural Information Processing Systems* (2006).

[94] Vintsyuk, T K. Speech discrimination by dynamic programming. *Cybernetics 4*, 1 (Apr. 1968), 52–57.

[95] Wang, Chang, and Mahadevan, Sridhar. Manifold alignment using Procrustes analysis. In *Proceedings of the International Conference on Machine Learning* (2008), pp. 1120–1127.

[96] Wang, Kongming, Begleiter, Henri, and Projesz, Bernice. Warp-averaging event-related potentials. *Clinical Neurophysiology*, 112 (2001), 1917–1924.

[97] Welling, Max, Hinton, Geoffrey E, and Osindero, Simon. Learning Sparse Topographic Representations with Products of Student-t Distributions. In *Advances in Neural Information Processing Systems* (2003).

[98] Williams, Adam, Ganesan, Deepak, and Hanson, Allen R. Aging in Place: Fall Detection and Localization in a Distributed Smart Camera Network. *ACM Multimedia* (Jan. 2007).

[99] Xi, Xiaopeng, Keogh, Eamonn, Shelton, Christian, Wei, Li, and Ratanamahatana, Chotirat Ann. Fast Time Series Classification Using Numerosity Reduction. In *Proceedings of the International Conference on Machine Learning* (2005).

[100] Yang, Jianchao, Yu, Kai, Gong, Yihong, and Huang, Thomas S. Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1794–1801.

[101] Zeiler, Matthew, Krishnan, Dilip, Taylor, Graham, and Fergus, Rob. Deconvolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[102] Zhou, Feng, and De la Torre, Fernando. Generalized Time Warping for Multimodal Alignment of Human Motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Apr. 2012).

[103] Zollei, Lilla, Learned-Miller, Erik G, Grimson, Eric, and Wells, William M. Efficient population registration of 3D data. In *IEEE ICCV Workshop on Computer Vision for Biomedical Image Applications: Current Techniques and Future Trends* (2005).